Autodesk®
Scaleform®

Getting Started with Scaleform Video

本書では、Scaleform 4.3 で Scaleform Video を使用する場合の詳細について説明しています。

著者: Matthew Doyle, Vladislav Merker
バージョン: 3.01
最終更新日: 2011 年 9 月 15 日

Autodesk®
GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.3

MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE REGARDING THESE MATERIALS.

Autodesk Scaleform の連絡先:

ドキュメント	Getting Started with Scaleform Video (Scaleform Video を使ったのクイック ガイド)
住所	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. はじめに

CRI™を備えた Autodesk® Scaleform® Video™は、Scaleform と完全に一体化した総合ビデオ ソリューションです。これは、ユーザーが高度なビデオ再生機能を、使用している Adobe® Flash®アセットに追加できるプレミアム モジュールで、統合する Flash ビデオ パイプラインを活用します。この新規ビデオ モジュールを使って、開発者はオープニングのロゴ、メイン メニュー、HUD、ゲーム内のテクスチャ、さらにフルスクリーンの映画のようなカット シーンの高解像度でノイズのない映像を再生することができます。

既存のビデオ コーデックよりも優れた再生機能とエンコード機能のため、CRI Movie コーデックが採用されました。CRI Movie の次世代再生エンジンは特にリアルタイム ゲーム システム用に構築され、最新のマルチコア ハードウェアの利点を生かしています。CRI Middleware は 1700 タイトル以上のゲームで使用されています。

Scaleform Video を使うと、PC やゲーム コンソール (Xbox 360®、PLAYSTATION®3 (PS3™)、Wii™) で、フルスクリーン、ウィンドウ モード、アルファ透明度など任意の解像度のビデオを再生できるようになります。これは、ストリーミングができるように基礎から設計され、完全にマルチスレッドです。Scaleform Video を使ってプラットフォームごとにさまざまな高解像度ビデオをエクスポートして、各プラットフォームに対して画質、フレーム レート、解像度、ビットレート、サイズ、縦横比、その他を最適化し、最善のビデオ品質とパフォーマンスを保証することができます。

Scaleform Video のワークフローを使うと、ユーザーは簡単に Adobe Premiere®、Adobe After Effects®などのビデオ編集アプリケーションで映像を作成し、Adobe Flash にインポートして、ゲームにエクスポートすることが簡単にできます。この過程で、字幕のセットアップ、5.1 オーディオ、他の言語用にローカライズされた吹き替え、さらにゲーム内のムービーをインタラクティブにするインタラクティブ キューの追加や、さらにそれ以上のことを簡単に行なうことができます。また、Scaleform Video はアルファ チャンネル、キュー ポイントなどの多くの高度な機能も備えており、開発者は自分のインターフェイスやカットシーンだけではなくそれ以上にビデオを活用することができます。

Scaleform Video SDK には以下のアイテムが含まれています:

- Scaleform Video ランタイム ライブラリ
- Scaleform Video ツール
- チュートリアル/サンプル
- Getting Started with Scaleform Video (本書)

日本のお客さまへ：
日本では、CRI Movie コーデックは Scaleform Video モジュールに含まれていません。Scaleform Video の機能をお使いになるには、株式会社 CRI・ミドルウェアから
CRI Movie™ for Scaleform を別途ライセンスしていただく必要があります。この点ご留意ください。

2. Scaleform Video について

Scaleform Video は複数のプラットフォームをサポートする高解像度、高音質のビデオ再生システムです。各プラットフォームの特質を最大限に生かしながら、最高水準のグローバル スタンダードに準拠した高品質のムービーの作成を支援します。また、多言語の音声や字幕など、多様な地域で販売されるゲームに高度な機能も提供します。

2.1 インストール先

Scaleform Video はビデオ エンコーダを以下の場所にインストールしています:

C:\Program Files\Scaleform\GFx SDK 4.3\Bin\Tools\VideoEncoder

Scaleform Video はビデオ デモを以下の場所にインストールしています:

C:\Program Files\Scaleform\GFx SDK 4.3\Bin\Data\AS2\Video

このデモには 3 つの異なる ビデオ プレーヤーを持つウィンドウが含まれ、それぞれで別のビデオ ファイルを再生することができます。このディレクトリには、テストに使用できるあらかじめエンコードされた USM ファイルの他に、デモに関連する Flash ファイルや Video ファイルも含まれています。Scaleform Player で *videodemo.swf* ファイルを起動して、デモを試してみてください。

2.2 チュートリアル ファイル

このチュートリアルで使用するサンプル ファイルが、Encoder と同じディレクトリに含まれています。チュートリアルの手順に従って作業するとき、これらのサンプルを使用します。

- *getting_started_with_video_tutorial.fla* – これはオリジナルの Flash ファイルで、このチュートリアルで説明するものすべてが含まれています。
- *getting_started_with_video_tutorial.swf* – これは Flash ファイルのパブリッシュされたバージョンです。
- *sample.avi* – エンコーディング用のサンプル ビデオです。
- *sample_audio.wav* – エンコーディング用のサンプル オーディオです。
- *sample_cue_points.txt* – エンコーディング用のサンプル キュー ポイントです。
- *sample_subtitles.txt* – エンコーディング用のサンプル字幕です。

2.3 Encoder ディレクトリ

Scaleform Encoder ディレクトリには、その操作に不可欠な複数のファイルが含まれています。

- Medianoche.exe — これはコマンド ライン エンコーダです。
- ScaleformVideoEncoder.exe — これはエンコーダのグラフィカル フロント エンドです (本書で説明します)。
- TMPGLib.dll — これはエンコーダの心臓部です。
- VideoEncoderUtil.dll — ScaleformVideoEncoder.exe に必要なライブラリです。
- VideoPlayer.swf — Scaleform Video Encoder ウィンドウで [Preview (プレビュー)] を押したときに、出力されるビデオ ファイルを制御できるグラフィック ビデオ プレーヤーです。

2.4 利点

- 高画質のムービー再生
  簡単な API を使って、高画質のムービーを再生することができます。フルハイビジョン (HD) 解像度 (1080p など) でムービー再生も可能です。

- 多言語音声と字幕のサポート
  多言語の音声と字幕を 1 つのムービーに入れることができます。これは複数の地域で販売されるゲームの総開発コストを削減します。

- 簡単な操作でプロレベルのエンコーディング
  使いやすいエンコード ツールがこの SDK パッケージに添付されています。

2.5 機能

CRI Movie™を備えた Scaleform® Video は、開発者がハイ パフォーマンスのビデオ再生機能を、自らのインタラクティブ コンテンツに Adobe Flash®を使って追加することができるプレミアム モジュールです。Scaleform Video を使うと、非常に高画質、高解像度でノイズのないビデオを、さまざまな次世代のプラットフォーム上で再生することができます。フルスクリーン、ウィンドウ モード、3D ゲーム エンジンのテクスチャ、または透明度付きでビデオの再生を行います。オープニング ロゴ、メイン メニュー、ゲーム内の HUD、ゲーム内のテクスチャ、ゲーム内のビデオ スクリーン、フルスクリーンの映画のようなカット シーン、スクリーンのローディングなどを含むさまざまなアプリケーションのための自身のインタラクティブ コンテンツで Video を使用します。Scaleform Video はすでに完全に Scaleform 4.1 と一体化しているので、ゲーム内のビデオの再生に必要な追加作業はほとんどありません。Scaleform Video は有名な受賞経験もある CRI Movie™コーデックに基づいていますが、大幅にワークフローが強化され、Adobe Flash と完全に一体化しています。

2.6 技術的な特長

- ビデオとオーディオのフォーマット: ほとんどの一般的に使用されているビデオとオーディオのフォーマット (AVI や WAV など) はこのエンコーダでサポートされています。
- マルチプラットフォームのサポート: Scaleform Video は Xbox 360、PS3、PC、Wii を含むすべての主要プラットフォーム上でサポートされ、プログラマはハードウェア特有の依存性に悩む必要がなくなります。
- さまざまなプラットフォームに対する最適のエンコーディング: エンコード中は、プラットフォームの特性やビデオの品質/圧縮率を考慮した最適のパラメータが適用できます。
- 字幕: 同じムービーに複数の字幕トラックがサポートされます。これによりユーザーは、個別のファイルに複数の言語の字幕トラックを作成することができ、それぞれの字幕の開始時間と終了時間が異なるようにすることができます。多言語のテキストは自動的に、Scaleform のフォント構成システムを使って、正しくローカライズされます。
- アルファ チャンネルのサポート: ユーザーは、ピクセルごとの透明度とムービー全体の透明度を可能にする埋め込まれたアルファ チャンネルで、自分のムービーに「仮想透明性」を作成することができます。これによりユーザーは多種多様のクリエイティブな効果を作り出すことができます。
- Render to Texture: ムービーは 3D の面に描画できるゲーム内のテクスチャ上でレンダリングすることができ、ユーザーはビデオを直接、そのゲームに統合することができます。
- オーディオ トラック: ビデオごとに複数のオーディオ トラック (32 まで) がサポートされます。これによりユーザーは多言語の吹き替えが可能になります。
- オーディオ トラックの切り替え: ユーザーは Scaleform 4.3 Flash エクステンションを使って、ActionScript™でさまざまなオーディオ トラックを切り替えることができます。
- ローカライズのサポート: ActionScript で実行時にオーディオ トラックや字幕トラックを切り替える機能により、Scaleform Video でローカライズがサポートされます。これは、さまざまな言語の顧客に対して UI ビデオを簡単に組織して維持することが可能な、大変便利な新機能です。
- サラウンド サウンドのサポート: 各オーディオ トラックはモノラル/ステレオ、または 5.1ch サラウンド サウンドにすることができます。
- 多言語のサラウンド サウンド: サラウンド サウンドのセンター チャンネル データは音声トラックで置き換えることができます。この機能を使うと、ユーザーはサラウンド オーディオ部材で簡単に音声トラックを再生することができます。
- 検索機能: Scaleform Video を使うと、ユーザーはビデオ ストリームの特定の時間を検索することができます。この検索値はそのストリームの開始時点に対して、または現在の位置に対して指定することができます。ユーザーは巻き戻しや早送りを行うことができます (正数と負数の検索値を使用します)。

- キュー ポイント: キュー ポイントはビデオ クリップ内で起きる重要な瞬間のことです。キュー ポイントを使うとビデオ クリップのさまざまなセグメントにアクセスすることができます。Scaleform Video Encoder を使って、USM ファイルに直接キュー ポイントを埋め込むことができます。このキュー ポイントは Flash のキュー ポイントとして使用することができます。そのビデオに埋め込まれたキュー ポイントに到達すると、ActionScript イベントがトリガされ、そのキュー ポイントに関連するすべての情報がユーザーに渡されます。これによりナビゲーション メニューのセットアップ、ビデオ内の特定のイベントの検索、インタラクティブなビデオの作成が可能になります。

2.7 ファイルを再生する

Scaleform Video はビデオ ムービーを USM ファイル拡張子を持つ Scaleform CRI Video フォーマットに変換します。USM ビデオ ファイルを再生するには、デスクトップ上にインストール済みである Scaleform Media Player のアイコンにそのファイルをドラッグ&ドロップするだけです。または、Scaleform Player を開いて、そのウィンドウに USM をドラッグ&ドロップします。

C:\Program Files\Scaleform\GFx SDK 4.3\Bin\Data\AS2\Video\ scaleform_logo.usm を Player にドラッグ&ドロップして、動作を確認してみてください。

2.8 ワークフロー

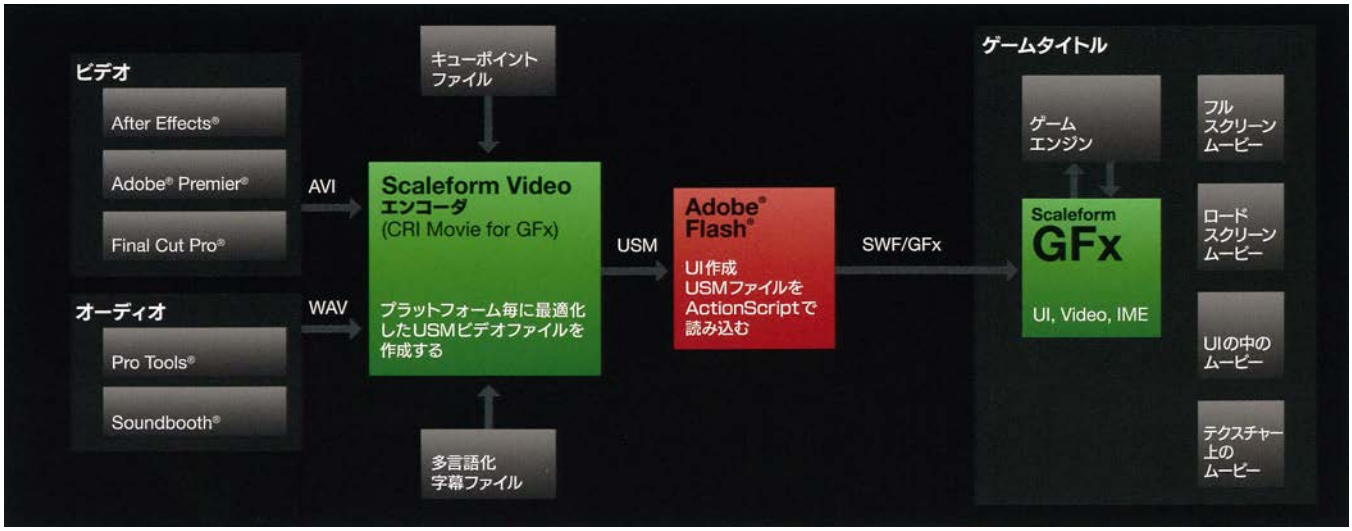

図 1: Scaleform Video のワークフロー

ゲーム内のビデオの作成に必要なワークフローは以下のとおりです:

1. Scaleform Video API をゲーム エンジンに統合する
2. Adobe Premiere やその他のビデオ エンコード ソフトウェアから、ビデオを AVI フォーマットにエクスポートする
3. Scaleform Video Encoder を使って AVI ビデオを USM フォーマットに変換する
4. USM にエンコードしたビデオを組み込む Adobe Flash SWF ファイルを作成する
5. ビデオ、字幕、キュー ポイントなどを制御する ActionScript を設定する
6. パブリッシュした SWF ファイルをゲームにインポートする
7. ゲーム エンジンで使用可能な方法を使って、ゲームでその SWF ファイルを再生する

*注意: USM ビデオを Flash ファイルに埋め込むのではなく、ActionScript を使ってそれを参照するようにしてください。ビデオをセットアップしてゲームで再生すると、そのビデオをハード ドライブ上の同じ物理的な場所とファイル名に再エンコードすることで、Flash ファイルで何も更新せずにそのビデオで反復できるようになります。

3. クイック ガイド: Scaleform のビデオをエンコードする

Scaleform 4.3 で使用するビデオは、CRI の USM フォーマットにエンコードしなければなりません。CRI のフォーマットは、ハイビジョン ディスプレイの最高の画質とパフォーマンスで実行する必要のあるゲームに高度に最適化されています。これを簡単に行うために、Scaleform は Scaleform Video Encoder を提供しています。この章では、Encoder を使って USM をエンコードする方法をステップごとに説明し、キュー ポイント、字幕、オーディオの統合に関して詳しく紹介しています。

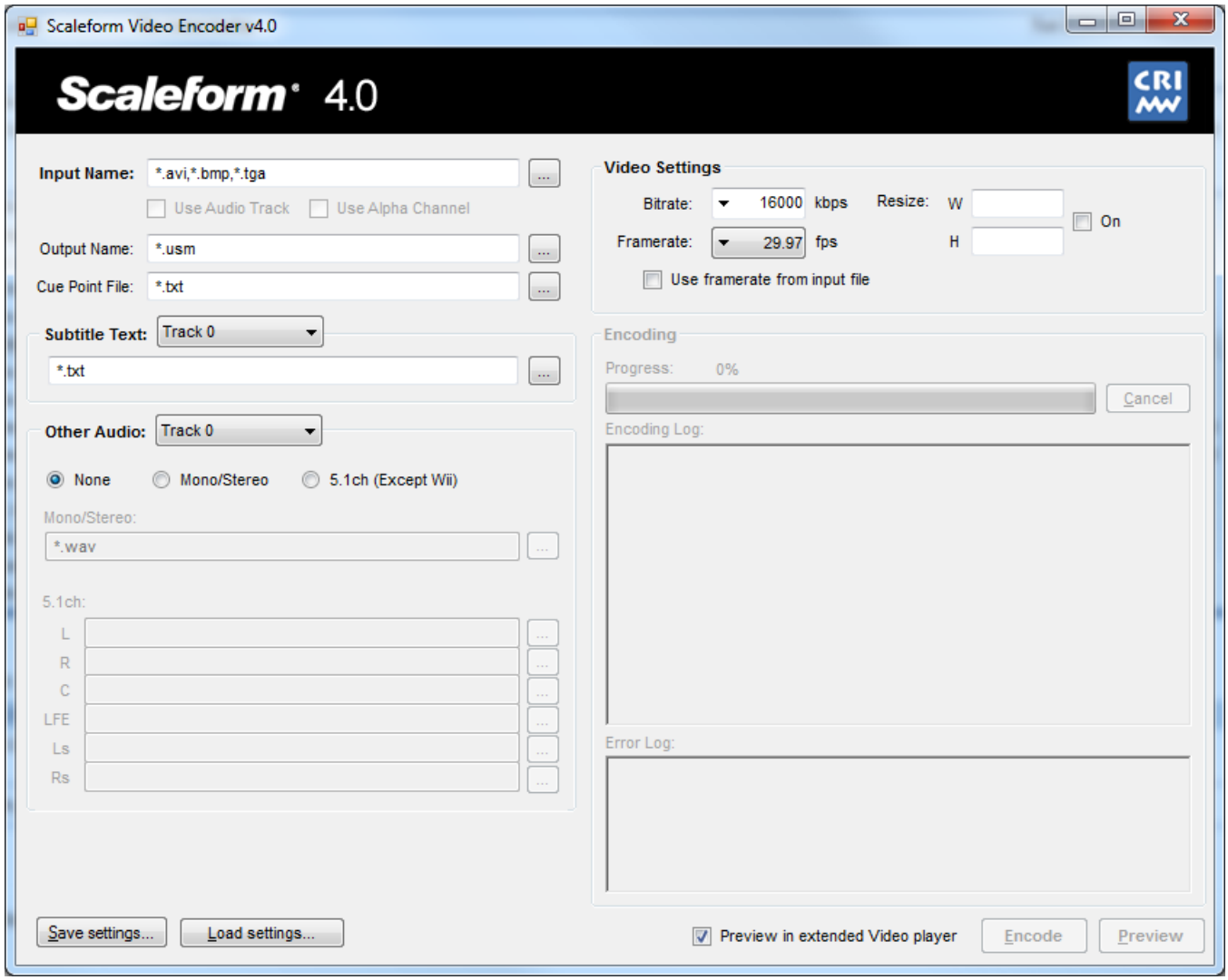

図 2: Scaleform Video Encoder

3.1 基本的なエンコーディングーステップごとの手順

ビデオを USM フォーマットに変換するプロセスは簡単なものです。字幕やキュー ポイントなどの一部の手順はオプションとなります。このチュートリアルでは、アルファ チャンネル (ピクセルごとの透明度)、キュー ポイント、字幕、ステレオ (2 チャンネル) 音声を備えた USM ビデオの作成に必要な手順を紹介します。

Windows™の [スタート] メニューのショートカットを使って、Scaleform Video Encoder を起動します。Scaleform Video のデフォルトのインストールでは、Encoder は以下の場所にあります:

Windows Vista:[スタート]->[すべてのプログラム]->[Scaleform]->[GFx SDK 4.3]
->[Video]
Windows XP:[スタート]->[すべてのプログラム]->[Scaleform]->[GFx SDK 4.3]->[Video]

1. [ブラウズ] ボタン  を使って、変換する AVI フォーマットのビデオをロードします。
    a. ビデオは AVI、または連番付きの一連の BMP あるいは TGA の場合があります。
    b. 連番付きの BMP か TGA の場合、その中の最初のファイルを選択します。

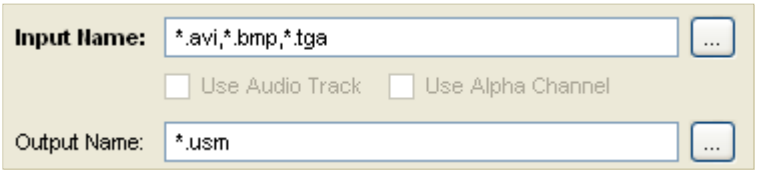

図 3: エンコードするビデオ ファイルを [Input Name] フィールドに入力する

2. そのビデオに内蔵のオーディオ トラックがある場合は、[Use Audio Track (オーディオ トラックを使用する)] をオンにします。AVI ファイルの場合、このオプションはデフォルトでオンになります。BMP と TGA の場合はオフになります。
3. 最終の USM ビデオにエンコードされるアルファ チャンネルがビデオにある場合、[Use Alpha Channel (アルファ チャンネルを使用する)] をオンにします。これにより、そのアルファ チャンネルが白の塗りつぶしではない場所では、下にあるコンテンツが透けて見えます。このオプションはデフォルトでオフになっています。
4. [Output Name (出力名)] フィールドにはデフォルトでソース ビデオのパスが表示されますが、必要であれば、[ブラウズ] ボタンで別のフォルダを指定して、エンコードした USM を保存します。
5. [Encode (エンコード)] ボタンをクリックしてエンコーディングを開始します。

3.2 最初のエンコード サンプル

1. Encoder を起動します。

2. *C:\Program Files\Scaleform\GFx SDK 4.3\Bin\Data\Tools\VideoEncoder\sample.avi* をロードします。
3. [Encode (エンコード)] をクリックして、デフォルト設定でサンプル ムービーのエンコードを開始します。
4. エンコードが問題なく終了したら、[Preview (プレビュー)] ボタンが使用できるようになります。このボタンをクリックして、USM にエンコードされたビデオを確認します。

3.3 キュー ポイント

キュー ポイントはチャプタ ナビゲーションのセットアップや、SWF のデータの変更、またはイベントの発生に便利です。エンコードされるビデオにキュー ポイントがない場合、次の章に進んでもかまいませんが、この章のキュー ポイントの説明に目を通すことをお勧めします。

最終的にエンコードされた USM がキュー ポイントを使用するには、[Cue Point File (キュー ポイント ファイル)] フィールドの隣の [ブラウズ] ボタンをクリックして、キュー ポイント テキスト ファイルを探します。

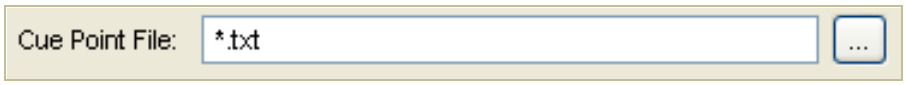

図 4: キュー ポイント テキスト ファイルを検索する

キュー ポイント テキスト ファイルには、各キュー ポイントが一行ずつリストアップされています。ファイルの一行目には、ミリ秒単位の表示間隔 (通常は 1000 ミリ秒) が含まれているはずです。この間隔は各キュー ポイントの時間値を決定する際のベースとして使用されます。5000 という値は 5 秒 (5000 / 1000 = 5) です。この間隔の後に続く行は、それぞれがエンコードされたビデオで使用する個別のキュー ポイントを含んでいます。

キュー ポイントにはナビゲーションとイベントの 2 種類があります。ナビゲーション キュー ポイントはチャプタの選択に使用されます。DVD のチャプタ メニューとほぼ同じです。イベント キュー ポイントはパラメータの設定に使用されます。各イベント キュー ポイントは複数のパラメータを持つ場合があります。パラメータはキー=値として一覧表示されます。1 つのキュー ポイントにカンマで区切った複数のパラメータが表示される場合もあります。Flash ファイルでキュー ポイントを使用する方法の詳細については、「ActionScript でビデオを操作する」を参照してください。

一行のキュー ポイント形式

```
time, cue point type (0 or 1), cue point name, parameter1, parameter2, …
paramter10
```

例:

```
1000, 0, cue_point_1, my_param=5, my_other_param=10
```

サンプルの CuePoint.txt ファイルの内容

```
1000
0,0,start
1000,0,cue1
2000,1,cue2,name1_0=value1_0,name1_1=value1_1
3000,0,cue3
4000,1,cue4,name4_0=value4_0
```

```
5000,0,cue5
6000,1,cue6,name6_0=value6_0,name6_1=value6_1
7000,0,cue7,name7_0=value7_0,name7_1=value7_1,name7_2=value7_2
8000,1,cue8,name8_0=value_0
9000,0,cue9
10000,1,cue10,name10_0=value10_0,name10_1=value10_1
11000,0,end
```

3.3.1 キュー ポイントを持つサンプル ビデオを再エンコードする

1. Scaleform Video Encoder を起動します。
2. *C:\Program Files\Scaleform\GFx SDK 4.3\Bin\Tools\VideoEncoder\sample.avi* をロードします。
3. [Cue Point File (キュー ポイント ファイル)] テキスト フィールドの隣の [ブラウズ] ボタンをクリックして、*sample_cue_points.txt* ファイルをロードします。このファイルは *sample.avi* と同じ場所にあります。
4. [Encode (エンコード)] をクリックして、デフォルト設定でサンプル ムービーのエンコードを開始します。
5. エンコード処理が問題なく終了したら、[Preview (プレビュー)] ボタンが使用できるようになります。このボタンをクリックして、USM にエンコードされたビデオを確認します。
6. Video プレーヤーの早送りと巻き戻しボタンを使って、キュー ポイントからキュー ポイントへ進めたり、戻したりします。

3.4 字幕

字幕のテキスト ファイルには、ビデオの各字幕のリストが一行ずつ含まれています。字幕はビデオ内で特定の時間にテキストを表示します。ビデオに字幕がない場合、次の章に進んでもかまいませんが、ビデオに字幕を追加する方法について理解するためにこの章を読むことをお勧めします。

複数の字幕ファイルが許可されます。字幕トラックごとに 1 ファイルです。これにより各言語の字幕ファイルを作成することができます。言語の中には、その言語の字幕の長さとその字幕を読む時間のために、他とは異なる開始時間と終了時間が必要なものもあります。

1. [Subtitle Text (字幕テキスト)] の隣にあるドロップ ダウン メニューからトラック番号を選択して、使用する字幕のトラックを選択します。最大で 32 個のトラックが使用できます (Track 0-31)。
2. [ブラウス] ボタンをクリックして、そのトラックで使用する字幕テキスト ファイルを選択します。

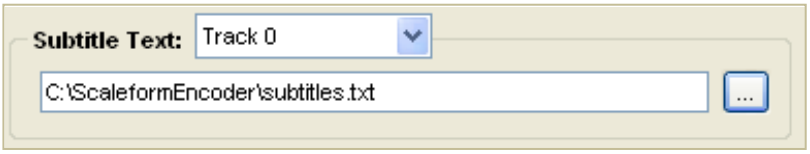

図 5: オーディオ トラックを選択しファイルを検索する

字幕ファイルの一行目には (コメント行以外)、ミリ秒単位の表示間隔 (通常は 1000 ミリ秒) が含まれているはずです。この値は字幕それぞれの開始時間と終了時間を決定するためのベースです。一行目の後に続く行には、開始時間、終了時間、メッセージ ID の 3 種類の情報が含まれています。メッセージ ID はコードで使用され、テキストの適切な文字列を表示します。あるいは、このパラメータを字幕として表示される実際の文字列として使用することもできます。Flash ファイルで字幕を使用する方法の詳細については、「ActionScript でビデオを操作する」を参照してください。

一行の字幕形式

```
start time, end time, messageID
```

サンプルの Subtitle.txt ファイルの内容

```
1000
2000, 5000, This is the first subtitle.
8000, 11000, subtitleId2
```

3.4.1 字幕を持つサンプル ビデオを再エンコードする

1. Scaleform Video Encoder を起動します。
2. *C:\Program Files\Scaleform\GFx SDK 4.3\Bin\Tools\VideoEncoder\sample.avi* をロードします。
3. [Cue Point File (キュー ポイント ファイル)] テキスト フィールドの隣の [ブラウズ] ボタンをクリックして、*sample_cue_points.txt* ファイルをロードします。このファイルは *sample.avi* と同じ場所にあります。
4. [Subtitle Text (字幕テキスト)] ドロップダウンのすぐ下にあるテキスト フィールドの隣の [ブラウズ] ボタンをクリックして、*sample_subtitles.txt* ファイルをロードします。このファイルは *sample.avi* と同じ場所にあります。
5. [Encode (エンコード)] をクリックして、デフォルト設定でサンプル ムービーのエンコードを開始します。
6. エンコード処理が問題なく終了したら、デスクトップのショートカット アイコンをクリックして Scaleform プレイヤー（Scaleform Media Player）を起ち上げ、そこに *getting_started_with_video_tutorial.swf* ファイルをドラッグ&ドロップしてください。
7. Video プレーヤーでは、字幕はそのビデオの一番下に表示されるので注意してください。字幕は 2-5 秒で表示され、再度 8-11 秒で表示されるはずです。

3.5 オーディオ

Encoder のオーディオ チャンネル セクションでは、モノラル/ステレオ、または 5.1 チャンネル サラウンド サウンドの複数のオーディオ トラックを追加することができます。チャンネルにはそれぞれ異なるオーディオ ファイルを含めることができます。このファイルは多言語の音声トラックのエンコードに便利です。

1. まず、Scaleform Video Encoder の [Other Audio (その他のオーディオ)] の隣にあるドロップダウン メニューからオーディオ トラックを選択します。
2. オーディオの種類を [None (なし)]、[Mono/Stereo (モノラル/ステレオ)]、[5.1ch (Except Wii) (5.1ch (Wii を除く))] から選択します。

   注意: 名前から分かるように、5.1 は任天堂の Wii では使用できませんが、その他のサポートされているプラットフォーム上では動作します。
3. [Mono/Stereo (モノラル/ステレオ)] の場合、[Mono/Stereo (モノラル/ステレオ)] テキスト フィールドの隣の [ブラウズ] ボタンを使用して、WAV フォーマットのソース オーディオ ファイルを探します。
4. [Use Audio Track (オーディオ トラックを使用する)] がオンになっている場合、ソース ビデオ ファイルへのパスがすでにこのフィールドに表示されています。

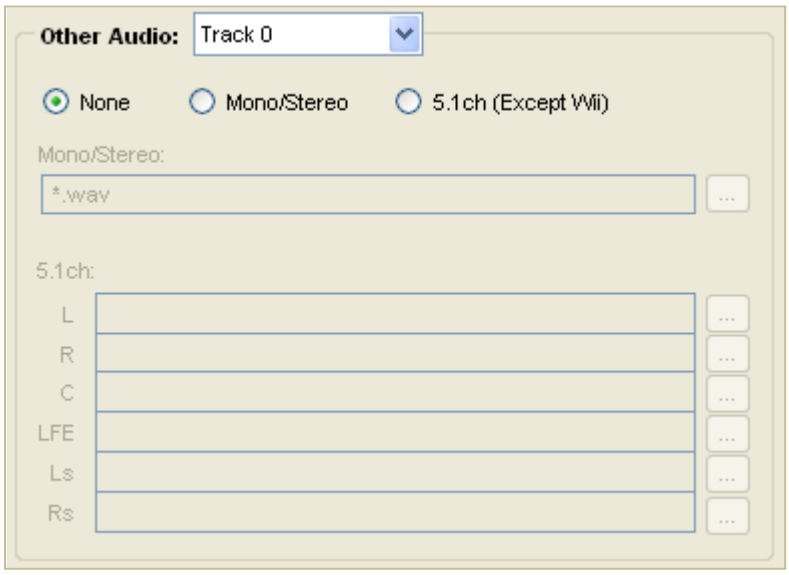

図 6: オーディオ トラックを選択しファイルを検索する

5. 5.1ch オーディオの場合、各チャンネル L、R、C、LFE、Ls、Rs の隣にある [ブラウズ] ボタンを使って、チャンネルごとにソース オーディオファイルをロードします。

5.1 チャンネル サラウンド サウンドで使用するチャンネル:

L – 左のスピーカー

R – 右のスピーカー

C – センター スピーカー (主に音声に使用されます)

LFE – サブウーファー

Ls – リア左のサラウンド サウンド スピーカー

Rs – リア右のサラウンド サウンド スピーカー

3.5.1 2 つのオーディオトラックを持つサンプル ビデオを再エンコードする

1. Scaleform Video Encoder を起動します。
2. *C:\Program Files\Scaleform\GFx SDK 4.3\Bin\Tools\VideoEncoder\sample.avi* をロードします。
3. [Cue Point File (キュー ポイント ファイル)] フィールドの隣の [ブラウズ] ボタンをクリックして、*sample_cue_points.txt* ファイルをロードします。このファイルは *sample.avi* と同じ場所にあります。
4. [Subtitle Text (字幕テキスト)] ドロップダウンのすぐ下にあるテキスト フィールドの隣の [ブラウズ] ボタンをクリックして、*sample_subtitles.txt* ファイルをロードします。このファイルは *sample.avi* と同じ場所にあります。
5. [Other Audio (その他のオーディオ)] ドロップダウンを [Track 1] に設定します。
6. [Mono/Stereo (モノラル/ステレオ)] ラジオ ボタンを選択します。
7. [Mono/Stereo (モノラル/ステレオ)] テキスト フィールドの隣にある [ブラウズ] ボタンをクリックして、*sampe_audio.wav* ファイルをロードします。このファイルは *sample.avi* ファイルと同じ場所にあります。
8. [Encode (エンコード)] をクリックして、デフォルト設定でサンプル ムービーのエンコードを開始します。注意: ビデオの長さがオーディオの長さよりも長いという警告メッセージが表示される場合があります。このメッセージは無視してください。エンコード 処理はそれでも順調に進みます。
9. エンコード処理が問題なく終了したら、[Preview (プレビュー)] ボタンが使用できるようになります。このボタンをクリックして、USM にエンコードされたビデオを確認します。
10. Video プレーヤーの右下、[Subtitle (字幕)] コンボ ボックスのすぐ上にある [Audio Channel (オーディオ チャンネル)] コンボ ボックスの左右の矢印ボタンをクリックして、オーディオ チャンネルを切り替えます。

3.6 ビデオ設定

このビデオ設定の章では、最終的なビデオの希望する画質レベル、圧縮、さらにファイル サイズを、1 つの設定 [Bitrate (ビットレート)] で設定し、さらにビデオのフレームレートと最終的な幅と高さを設定します。

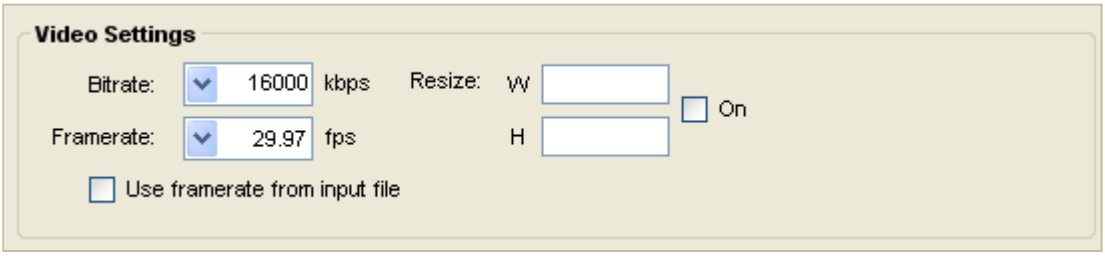

図 7: ビデオ設定

1. [Bitrate (ビットレート)] の上下の矢印を使用するか、値を入力して希望するビデオの画質を設定します。ビットレートはキロビット/秒単位です。これはそのビデオで使用される圧縮の程度を決定します。ビットレートが高いほど、ビデオ ファイルの画質は良くなり、ファイル サイズは大きくなります。ビットレートが低いほど圧縮率は大きくなるので、画質は低下しファイル サイズは小さくなります。

   以下の表はプラットフォーム別の最適なエンコード率を表しています。多くの場合、最初はエンコード処理を速くするために低い解像度と高いビットレートでエンコードし、その後、ビデオの最終バージョンが用意できたら、以下の表のいずれかのレートで高い解像度のビデオを作成するのが最善の方法です。

各プラットフォームの推奨ビットレート (kbps)					
ビデオ フォーマット		PS3	Xbox360	Wii	PC
1080p	高画質	40000	40000	使用不可	40000
1080p	標準	30000	30000	使用不可	30000
1080p	高圧縮	20000	20000	n/a	20000
720p	高画質	36000	36000	n/a	36000
720p	標準	26000	26000	n/a	26000
720p	高圧縮	16000	16000	n/a	16000
480p	高画質	16000	16000	8000	16000

480p	標準	12000	12000	6000	12000
480p	高圧縮	8000	8000	4000	8000

2. (オプション) [Framerate (フレームレート)] ドロップダウン メニューを使って、エンコードされたビデオのフレームレートを調整します。これはデフォルトでは 29.97 fps の標準の NTSC ビデオに設定されています。AVI が Encoder にロードされると、[Use framerate from input file (入力ファイルのフレームレートを使用する)] がオンになり、Encoder はソース ビデオのオリジナルのフレームレートを使用します。最終的な出力フレームレートを操作したい場合、このオプションをオフにしてドロップダウンを使用します。フレームレートは 1 秒間に表示されるビデオのフレーム数の単位です。NTSC 標準のテレビ信号は 1 秒間に 29.97 フレームで送信され、これは推奨設定です。PC のモニタは通常、1 秒間に 60 フレーム、または 60 ヘルツで同期されます。

3. (オプション) [Resize (サイズ変更)] の [W (幅)] と [H (高さ)] テキスト フィールドを使って、エンコードされたビデオの最終的な出力サイズを調整します。幅を [W] に高さを [H] に入力して、[On] をオンにします。

4. [Encode (エンコード)] をクリックします。エンコードされたら、[Preview (プレビュー)] をクリックして、ビデオを再生します。

   ビデオのエンコードによって、出力される USM ビデオ ファイルと同じディレクトリにバッチ ファイルが作成されます。このバッチ ファイルには、このコマンド ライン エンコーダを使ったビデオのエンコードに必要なコマンドが含まれています。

5. エンコード処理が終わったら、[Preview (プレビュー)] をクリックして、ビデオを再生します。

   [Encode (エンコード)] ボタンの隣にある [Preview in extended Video player (拡張した Video Player でプレビューする)] をオフにして、再生設定なしでビデオをプレビューすることもできます。それ以外の場合は、このオプションはオンのままにしておきます。

4. クイック ガイド: ビデオを Flash に追加する

注意: この章は Flash オーサリング環境に精通していること、および USM にエンコードされたビデオ ファイルがすでに作成済みであると仮定しています。この章は ActionScript を使ってビデオを SWF に追加するプロセスについて、ステップごとに紹介します。この章は概要説明を意図したものなので、再生、一時停止、巻き戻し、ビデオ サイズの調整など、ビデオの制御の追加についての詳細は含まれていません。このような設定の詳細については、Flash のヘルプで NetStream クラスを参照してください。

ビデオを追加するには、まず始めに Scaleform Encoder で USM をエンコードします。この手順の詳細については、「クイック ガイド: Scaleform のビデオをエンコードする」を参照してください。この章の手順に従ってエンコードされた USM ビデオ ファイルを SWF ファイルに追加して、その SWF ファイルとビデオを Scaleform Player で再生します。

4.1 Flash でビデオをセットアップする

空の AS 2.0 Flash ファイルを作成して、まずはそのファイルを Scaleform Video Encoder ディレクトリに保存します。簡単にするため、このチュートリアルが使用する USM ビデオ ファイルは、パブリッシュされた SWF ファイルと同じディレクトリに入れます。

1. タイムラインの [レイヤー 1] という単語をダブルクリックして、'mvplayer' に名前を変更します。
2. [レイヤーの追加] ボタンを使って新規レイヤーを作成し、'actions' という名前に設定します。

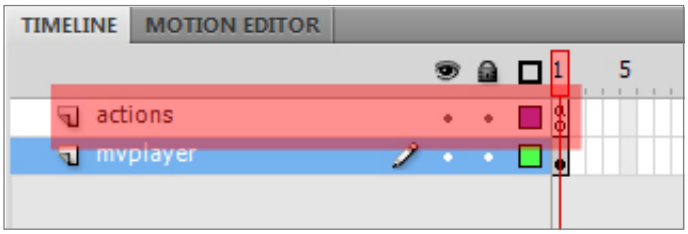

図 8: タイムライン上の actions レイヤーと mvplayer レイヤー

3. タイムライン上でこの *actions* レイヤーのフレーム 1 を選択し、F9 キーを押して [アクション] パネルを開き以下のコードを入力します:

```
_global.gfxExtensions = true;
_focusrect = false;
onLoad = function()
{
      mvplayer.playVideo("sample.usm");
}
```

注意: ビデオ ファイルがパブリッシュされた SWF ファイルと同じディレクトリに保存されている場合、単純にビデオ ファイル名を USM 拡張子を付けて入力します。ただし、ビデオ ファイルが他の場所に保管されている場合、そのファイルへの完全修飾パスを入力します。バックスラッシュをスラッシュに必ず置き換えてください。
例: *"C:/pathtovideo/sample.usm"*

4. タイムライン上の *mvplayer* レイヤーを選択します。
5. ステージ上にムービー クリップを作成します。
    a. この方法の 1 つとして、外枠が黒で透明の塗りつぶしの大きなボックスをステージをほとんど覆うように描画します。
    b. 外枠の任意の場所でダブルクリックして、ボックス全体を選択します。
    c. 外枠の任意の場所にマウス カーソルを置いて右クリックして、[シンボルに変換] を選択します。

d. この新規ムービー クリップの名前を ‘mvplayer’ に設定して、[OK] をクリックします。

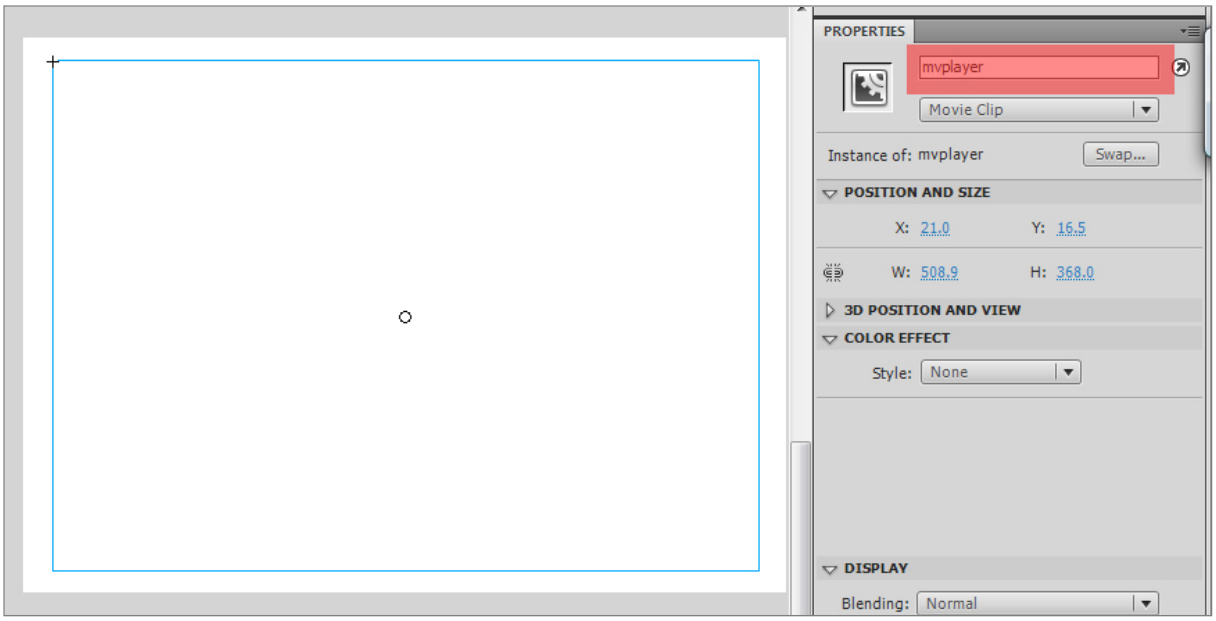

図 9: ムービー クリップ mvplayer を作成する

6. 外枠上をクリックして、ステージで新規の *mvplayer* ムービー クリップを選択します。
7. [プロパティ] タブでそのムービー クリップのインスタンス名を ‘mvplayer’ に変更します。
8. *mvplayer* ムービー クリップをダブルクリックしてそのタイムラインを表示します。
9. *mvplayer* ムービー クリップ内で、新規のレイヤーを作成して ‘actions’ というラベルを付けます。
10. タイムライン上でこの *actions* レイヤーのフレーム 1 を選択して、[アクション] パネルで以下のように入力します:

```
nc = new NetConnection();
nc.connect(null);
ns = new NetStream(nc);
video.attachVideo(ns);
this.attachAudio(ns);
var audio_sound:Sound = new Sound(this);
function playVideo(videoToPlay:String):Void
{
      ns.play(videoToPlay);
}
```

注意: 上記のコードを入力する前に、レイヤー1 のグラフィック イメージではなく、*mvplayer* タイムラインの actions レイヤーのフレーム 1 が選択されていることを確認してください。

11. [ライブラリ] パネルで右クリックして、ポップアップ メニューから [新規ビデオ] を選択します。

12. この新規ビデオに ‘video’ という名前を付け [OK] をクリックします。

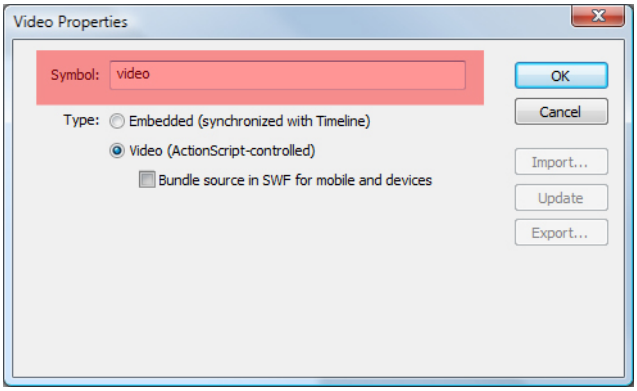

図 10: ビデオ オブジェクトを作成する

13. [レイヤーの追加] ボタンを使って新規レイヤーを作成します。このレイヤーは「レイヤー3」という名前になるはずです。

14. 「レイヤー3」のフレーム 1 で、この新規ビデオ オブジェクトを [ライブラリ] パネルからステージにドラッグし、必要に応じて位置を変え、サイズを変更します。

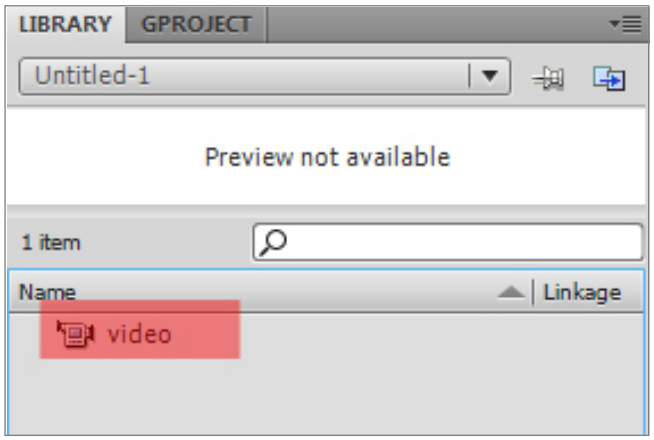

図 11: [ライブラリ] パネルのビデオ オブジェクト

15. ステージ上のこのビデオ オブジェクトを選択して、[プロパティ] タブでそのインスタンス名を ‘video’ に設定します。

16. ムービーを保存しパブリッシュします。

4.2 ビデオをテストする

ビデオ付きの SWF をいったん作成すると、Scaleform Player でそのビデオを表示するのは簡単です。

1. Flash の [Scaleform Launcher] パネルからその SWF を起動します。

または、

1. Scaleform Player ウィンドウを開きます。
2. その SWF ファイルを Scaleform Player ウィンドウにドラッグ&ドロップします。
    a. または、SWF ファイルを Scaleform Player ショートカットにドラッグします。

5. クイック ガイド: ActionScript でビデオを操作する

完全にエンコードした USM (字幕、キュー ポイント、複数のオーディオ チャンネル付き) を作成し、ActionScript を使ってそのビデオを Flash ファイルに追加した後で、その字幕の表示の制御、オーディオ チャンネルを使ったサウンド出力の変更、さらにイベントをナビゲートし実行するキュー ポイントの使用が可能になります。注意: この章は「クイック ガイド: ビデオを Flash に追加する」のチュートリアルを元にして説明します。このチュートリアルを使用するには、前章で説明した Flash ファイルを作成してください。

この章では、Flash SWF で Scaleform ビデオのキュー ポイント、字幕、オーディオ トラックにアクセスして使用するプロセスを紹介します。このチュートリアルでは、ActionScript でビデオの作業を行うための基礎を説明する予定です。提供されているコード サンプルの一部は推奨コードですが、製品を完成するための唯一の手段というわけではありません。最終的に、特定のプロジェクトのニーズによって、コードのインプリメンテーションは左右されます。

5.1 ActionScript でキュー ポイントを操作する

USM ファイルはキュー ポイント付きでエンコードされ、ナビゲーション ポイントやイベント ポイントを備えたビデオが可能になります。ナビゲーション ポイントは基本的なチャプタ ナビゲーションに役立ちます。イベント ポイントは Flash ファイルにキー/値パラメータを送信するのに有効です。Flash ファイルはイベントの発生やプロパティの変更で応答することができます。

注意: キュー ポイント テキスト ファイルと共にエンコードされた USM ビデオを備えた/再生する Flash SWF は、キュー ポイントにアクセスできる必要があります。本書の手順を順番に行った場合、すでに完全にエンコードされたキュー ポイント付きの sample.usm ビデオが手元にあるはずです。そうでない場合、「ビデオを Flash に追加する」と「キュー ポイント」の該当する箇所を参照してください。

キュー ポイントにアクセスするには、数行の ActionScript を追加しなければなりません。前章で作成した mvplayer ムービー クリップの actions レイヤーのフレーム 1 に、以下のコードを設定します。[アクション] パネルの最後の行にこのコードを入力、またはコピーします。

1. *mvplayer* ムービー クリップをダブルクリックします。
2. *mvplayer* の内部で、タイムラインの *actions* レイヤーの最初のキーフレームを選択して、以下のコード スニペットを [アクション] パネルのすでに存在するコードの下に入力します。
3. キュー ポイント イベントをリッスンして、リッスン時に `traceCuePoint` 関数を実行します。

```
ns.onCuePoint = traceCuePoint;
```

4. `traceCuePoint` 関数はパラメータとして、キュー ポイント オブジェクトを受け入れます。次に、この関数は `metaPropPJParams` オブジェクトにそのキュー ポイントのパラメータを保管します。そこから、この関数はキュー ポイント名 (`currentCuePoint.name`)、キュー ポイント時間 (`currentCuePoint.time`)、キュー ポイントの種類 (`currentCuePoint.type`) などのキュー ポイントの基本情報をトレース (出力ウィンドウに表示) します。このキュー ポイント データを使って、以下で説明するトレース インプリメンテーションの代わりに、ビデオの再生を (`ns.time` と連動して) 制御することができます:

```
function traceCuePoint(currentCuePoint:Object):Void
{
      var metaPropPJParams:Object = currentCuePoint.parameters;
      trace("\t\t name: " + currentCuePoint.name);
      trace("\t\t time: " + currentCuePoint.time);
      trace("\t\t type: " + currentCuePoint.type);
      if (metaPropPJParams != undefined)
      {
            trace("\t\t parameters:");
            traceObject(metaPropPJParams, 4);
      }
}
```

5. 最後に、キュー ポイントにパラメータがあれば、そのパラメータを以下の traceObject 関数を呼び出して、出力ウィンドウにトレースします。キュー ポイント データを使って SWF でプロパティ (変数) を変更したり、イベントを発生させることができます。

```
function traceObject(obj:Object, indent:Number):Void
{
	var indentString:String = "";
	for (var j:Number = 0; j < indent; j++)
	{
		indentString += "\t";
	}
	for (var i:String in obj)
	{
		if (typeof(obj[i]) == "object")
		{
			trace(indentString + " " + i + ": [Object]");
			traceObject(obj[i], indent + 1);
		}
		else
		{
			trace(indentString + " " + i + ": " + obj[i]);
		}
	}
}
```

6. [Scaleform Launcher] パネルを使って、または SWF ファイルを Scaleform Player ウィンドウにドラッグして、この Flash ファイルをパブリッシュし Scaleform Player でテストします。すべてが正確であれば、ビデオが再生されるときに、そのビデオのキュー ポイントは Scaleform Player の出力ウィンドウに表示されるはずです。

5.2 ActionScript で字幕を操作する

字幕付きでエンコードされたビデオの場合、字幕を Flash で操作して表示することができます。この字幕は映画の字幕と同じように動作します。ビデオで多言語の字幕をサポートすることができます。

注意: 字幕付きのビデオ ファイルを使用する前に、字幕テキスト ファイル付きでエンコードされた USM ビデオを備えた/再生する Flash SWF を作成しなければなりません。本書の手順を順番に行った場合、すでに完全にエンコードされた 2 つの字幕付きの sample.usm ビデオが手元にあるはずです。そうでない場合、「ビデオを Flash に追加する」と「字幕」の該当する箇所を参照してください。

1. *mvplayer* ムービー クリップをダブルクリックします。
2. *mvplayer* の内部で、タイムラインの *actions* レイヤーの最初のキーフレームを選択して、以下のコード スニペットを [アクション] パネルのすでに存在するコードの下に入力します。
3. まず、字幕チャンネルの数を保有する変数 `subtitleChannelNumber` を作成して、初期値を 0 に設定します:

```
var subtitleChannelNumber = 0;
```

4. 次に、ビデオ ファイルのメタデータから字幕チャンネルの数をリッスンします。注意: 字幕チャンネル 0 は字幕をオフにします。

```
ns.onMetaData = function(infoObject:Object)
{
      ns.subtitleTrack = 1;
      subtitleChannelNumber = infoObject.subtitleTracksNumber;
      trace("Subtitle Channels: " +  subtitleChannelNumber);
}
```

5. *mvplayer* ムービー クリップの内部でステージにダイナミック テキスト フィールドを作成して、インスタンス名を 'subtitle' に設定します。十分な大きさにして、目立つ色とフォントを選択してください。
6. 再生中にビデオが字幕が埋め込まれた時間に達すると、以下のコールバック関数が実行されます。`msg` という文字列は、その USM ムービーがエンコードされたときに、字幕テキスト ファイルに入力されていた文字列を含みます。最初の字幕の場合、この文字列は字幕ファイルに入力された通りに使用されます。変数名 `subtitleId2` の 2 つ目の字幕の場合、この変数名を使って、ActionScript で表示されるメッセージを決定します。`msg` の内容はステップ 5 で作成した *subtitle* ダイナミック テキスト フィールドに表示されます。

```
ns.onSubtitle = function(msg:String)
{
      trace("Subtitle: " + msg);
      subtitle.text = msg;
}
```

7. [Scaleform Launcher] パネルを使って、または SWF ファイルを Scaleform Player ウィンドウにドラッグして、この Flash ファイルをパブリッシュし Scaleform Player でテストします。すべてが正確であれば、ビデオが再生されるときに、出力ウィンドウと Flash ファイルのテキスト フィールドの両方でビデオの字幕が表示されるはずです。

5.3 ActionScript でオーディオ チャンネルを操作する

USM は複数のオーディオ チャンネル付きでエンコードされ、必要な言語ごとに個別のチャンネルを含むビデオを可能にします。SWF でこれらのオーディオ チャンネルにアクセスする場合、ほんのわずかなコードしか必要としない簡単な手順で行うことできます。

ビデオ ファイルのオーディオ チャンネルにアクセスするには、最低でも 1 つのオーディオ チャンネルと共にエンコードされた USM ビデオを備えた/再生する Flash SWF を作成する必要があります。この手順の簡単なチュートリアルについては、「ビデオを Flash に追加する」と「オーディオ」の該当箇所を参照してください。

mvplayer ムービー クリップの *actions* レイヤーのフレーム 1 に以下のコードを設定します。[アクション] パネルの最後の行にこのコードを入力、またはコピーします。

1. まだ表示されていない場合は、*mvplayer* ムービー クリップをダブルクリックしてそのタイムラインを表示します。
2. *mvplayer* の内部で、タイムラインの *actions* レイヤーの最初のキーフレームを選択して、以下のコード スニペットを [アクション] パネルのすでに存在するコードの下に入力します。
3. 2 つの変数を宣言します。最初の `AudioTracks` はビデオの各オーディオ トラックを保有する配列です。2 つ目の `CurAudioTrack` は現在選択されているトラックを保有します。この変数の初期値を 0 に設定します。

```
var AudioTracks:Array;
var CurAudioTrack = 0;
```

4. 次に、コールバック関数をビデオのメタデータをリッスンするように設定します。ビデオの `audioTracks` メタデータを `AudioTracks` 配列に割り当てます。

   注意: 字幕の制御を追加する章で ‘`ns.onMetaData`’ がすでに存在している場合、以下の ‘`AudioTracks = infoObject.audioTracks;`’ を、コードのそのブロック内部の ‘`trace("Subtitle Channels: " + subtitleChannelNumber)`の後、閉じ括弧 ‘}’ の前に追加します。

```
ns.onMetaData = function(infoObject:Object)
{
      AudioTracks = infoObject.audioTracks;
}
```

5. テスト用に、*mvplayer* ムービー クリップの内部で、ステージ上にボタンを表現する簡単な四角形のグラフィック ムービー クリップを作成します。そのムービー クリップに ‘prev_audiotrack’ というインスタンス名を付けます。

6. *mvplayer* の *actions* レイヤーの最初のキーフレーム上で、[アクション] パネルに以下のコードを入力します。このコードは prev_audiotrack ボタンを押したときに、オーディオ チャンネルを後方に反復します。

```
prev_audiotrack.onRelease = function()
{
      if (AudioTracks  == undefined || AudioTracks.length < 2)
      {
            return;
      }
      if (CurAudioTrack == 0)
      {
            CurAudioTrack = AudioTracks.length - 1;
      }
      else
      {
            CurAudioTrack--;
      }
      ns.audioTrack = AudioTracks[CurAudioTrack].trackIndex;
}
```

7. *mvplayer* 内部でステージ上にステップ 5 のムービー クリップを複製して、インスタンス名を 'next_audiotrack' に設定します。
8. *mvplayer* の *actions* レイヤーの最初のキーフレーム上で、[アクション] パネルに以下のコードを入力します。このコードは、next_audiotrack ボタンを押したときに、オーディオ チャンネルを前方に反復します。

```
next_audiotrack.onRelease = function()
{
     if (AudioTracks  == undefined || AudioTracks.length < 2)
     {
           return;
     }
     CurAudioTrack++;
     if (CurAudioTrack >= AudioTracks.length)
     {
           CurAudioTrack = 0;
     }
     ns.audioTrack = AudioTracks[CurAudioTrack].trackIndex;
}
```

9. この Flash ファイルをパブリッシュし Scaleform Player でテストします。すべてが正確であれば、[NEXT] や[PREV] オーディオ ボタンを押したときに、再生中のビデオのさまざまなオーディオ トラックがプレビュー表示できます (エンコードされた USM ファイルに複数のオーディオ トラックが含まれている場合。それ以外は 1 つのトラックしか使用できません)。

6. ActionScript のビデオ エクステンション

ビデオ ファイルの再生のために Scaleform がサポートする多くの ActionScript メソッドが、Flash には内蔵されています。さらに、Scaleform に特化した新規のエクステンションもあります。
'NetStream' を Adobe のヘルプ ファイルで検索して、内蔵の NetStream プロパティ、メソッド、イベントについて詳細を確認します。

6.1 サポートされる内蔵の *NetStream* プロパティ

```
currentFps:Number
```

表示する毎秒あたりのフレーム数です。

例:`var currentFps:Number = ns.currentFps;`

```
time:Number
```

秒単位の再生ヘッドの位置です。

例: `var currentTime:Number = ns.time;`

6.2 サポートされる内蔵の *NetStream* イベント

```
onCuePoint = function(infoObject:Object) {}
```

USM ファイルを再生中に、埋め込まれたキュー ポイントに達した場合に呼び出されます。

注意: 詳細については、Adobe Flash のヘルプで「onCuePoint」を検索してください。

```
onMetaData = function(infoObject:Object) {}
```

再生されている USM ファイルに埋め込まれた記述情報を Player が受け取ると呼び出されます。

注意: 詳細については、Adobe Flash のヘルプで「onMetaData」を検索してください。

Scaleform Extensions:

`audioTracks`

USM ファイルのオーディオ トラックの配列です。

例:

```
ns.onMetaData = function(infoObject:Object)
{
      audioTrackArray = infoObject.audioTracks;
}
```

`subtitleTracksNumber`

USM ファイルの字幕トラック数の合計です。

例:

```
ns.onMetaData = function(infoObject:Object)
{
      numSubtitleChannels = infoObject.subtitleTracksNumber;
      trace("Subtitle Channels: " +  numSubtitleChannels);
}
```

```
onStatus = function(infoObject:Object) {}
```

NetStream オブジェクトに対してステータスの変更やエラーが発生するたびに呼び出されます。

注意: 詳細については、Adobe Flash のヘルプで「onStatus」を検索してください。

6.3 サポートされる内蔵の NetStream メソッド

```
close()
```

そのストリーム上ですべてのデータの再生を停止して、ns.time プロパティを 0 に設定し、ストリームを別のことに使用できるようにします。

例: `ns.close();`

```
pause([flag:Boolean])
```

ストリームの再生を一時停止、または再開します。

例: `ns.pause(true);`

```
play(name:Object, start:Number, len:Number, reset:Object)
```

外部のビデオ (USM) ファイルの再生を開始します。

例: `ns.play("myVido.usm");`

```
seek(offset:Number)
```

ストリームの開始時点からの指定した秒数に最も近いキーフレームを検索します。

例: `ns.seek(50);`

```
setBufferTime(bufferTime:Number)
```

バッファするムービーデータの量を秒を単位として設定します。GFx::Video::Video はディスクから十分なデータをバッファするため、スムーズな再生が実現されると同時に、ディスクからの読み出し回数が少なくなります。バッファ サイズはビデオのビットレートや他のビデオ パラメータをもとに決定します。このバッファは、デフォルトでは、再生に有効な 1 秒分のデータを保持するだけの十分な大きさがあります。

例:`ns.setBufferTime(2.5);`

6.4 新規の Scaleform プロパティ

`subtitleTrack`

NetStream が現在使用している字幕のトラック番号です。

例: `ns.subtitleTrack = 1;`

`subtitleTrack = Number`

NetStream が現在使用している字幕のトラック番号です。

例: `ns.subtitleTrack = 1;`

`audioTrack = Number`

NetStream が使用するメインのオーディオ トラックです。ビデオファイルが複数のオーディオ トラックで構成される場合に（たとえば英語版とスペイン語版）、このプロパティを用いて特定言語のオーディオ トラックを設定します。

例:`ns.audioTrack = 2;`

`subAudioTrack = Number`

二次トラックまたは SubAudio トラックです。一般にダイアログ トラックの再生またはサウンド エフェクトとして、ムービーとともに NetSream が使用します。

例:`ns.subAudioTrack = 3;`

`voiceTrack = Number`

5.1 チャネル オーディオのセンター トラックを入れ替えます。5.1 チャネル ミュージックのボイス トラックを変更するときのみ使用します。

例:`ns.voiceTrack = 1;`

`setReloadThresholdTime(reloadTime:Number)`

リロード タイミングを設定します。ディスクからビデオ データ バッファにデータを補充する頻度を指定します。たとえば、アプリケーションがバッファ サイズを（setBufferTime を使って）4 秒に設定し、かつ、リロード スレッショルドを 1 秒に設定した場合、GFx::Video::Video は最初にバッファを 4 秒分の有効なデータで満たします。その 3 秒後には、有効なデータのデコードおよび消費が進んでバッファにはリロード スレッショルド 秒に満たないデータしか残っていない状態となるため、GFx::Video::Video はバッファを再補充します。

例:`ns.setReloadThresholdTime(1);`

`setNumberOfFramePools(numPools:Number)`

内部ビデオ バッファの個数を設定します。GFx::Video::Video は表示前のデコード済みフレーム バッファとして内部メモリを使用します。多くのフレームをプールできれば CPU 負荷が高い場合でもスムーズな再生が得られます。デフォルト値は 2 です。

例:`ns.setNumberOfFramePools(3);`

`setOpenTimeout(timeout:Number)`

ファイル オープンのタイムアウトを秒を単位として設定します（USM ヘッダのデコードに必要な時間）。デフォルト値は 5 です。この機能を無効（タイムアウト時間を無限に設定）するには 0 を渡します。

例:`ns.setOpenTimeout(5);`

`currentFrame:Number`

現在の再生ポジションを（フレームを単位として）返します。

例:`trace("Current FPS: " + ns.currentFps + " fps");`

`Loop = Boolean`

loop を true に設定すると、NetStream は loop が false に設定されるまでビデオを繰り返すように指示されます。

例: `ns.loop = true;`

`clipRect = new rectangle(x:Number, y:Number, width:Number, height:Number)`

ビデオの指定された四角形の領域のみを表示するために使用されます。

例: `ns.clipRect = new rectangle(100, 100,  240, 200);`

`onSubtitle = function(msg:String) {}`

Player が USM ファイルから字幕文字列を受け取ると呼び出されます。

例: `ns.onSubtitle = function(msg:String) { trace(msg); }`

7. ビデオの作成を理解する

ビデオ編集や変換の経験が少ないユーザーにとっては、ビデオの編集やビデオの作成は厄介な処理になる場合があります。ビデオの編集は、ビデオの再生品質やパフォーマンスに影響する多くの解像度や設定を行うことが可能です。ビデオの場合、ビデオをスムーズに最善の画質で再生できるように、解像度、縦横比、ビット レート、フレーム レート、さらにその他同様の要因を考慮することが一般的です。ハードウェアのプラットフォーム、使用事例、ゲーム エンジンはそれぞれ、最善のパフォーマンスを保証するために、ビデオのエンコーディングの方法がわずかに異なります。これにはエンコーディングの過程で、さまざまな設定で実験したりテストすることが必要になります。

例:単独のフルスクリーン ビデオ ムービーの再生は、比較的に簡単な条件しかありません。再生されるのはこれだけで、システム リソースをほとんど利用できるからです。ただし、複数のムービーのストリーミング、ムービーの背後でバックグラウンドの読み込み、ムービーのアルファ チャンネルの使用、テクスチャ上のムービーの再生、またはその他の高度な機能を使用するとき、正しい設定を探し出すにはさまざまなエレメントを使った実験が必要になる場合もあります。

フルハイビジョン 1080p のビデオ作成は、Xbox 360 や PS3™の最高品質のムービーのための最善の方法のように聞こえるかもしれませんが、ビデオを作成する前に複数の要因を検討しければなりません。まず、ファイル サイズが非常に大きなため、ハイビジョン ビデオの編集はきわめて困難で、遅々としたものになる場合があります。非常にハイエンドなマシンでビデオの編集を行っている場合は別ですが、ハイビジョンの編集は避けるべきです。ハイビジョンのビデオ、特に高ビットレート 1080p のビデオの画質はすばらしいものですが、同時にファイル サイズが非常に大きく再生に多くのシステム リソースを使うので、各ビデオに必要な解像度は早期に決定する必要があります。

1080p のビデオ ムービーを作成するかどうか決定するときに考慮するべきことは他にも、ビデオ ソース ファイルの種類、(動画、CG、ゲーム内など)、ビデオを操作する人物、予算、制作期間などが含まれます。

720p、1080p、さらにその他のハイビジョン解像度の実際の画質の違いを検証することが重要です。多くの場合、再生したときに見映えが十分であり、簡単に作成でき、小さい (ディスク領域をあまり必要としない)、編集が簡単で、ストリーミングの品質が良く、使用するメモリ量が少ないような、より低い解像度が存在します。ビデオの作成過程では、繰り返してテストを行い、ゲーム内でそのビデオがどのよう再生されるかだけではなく、パイプラインや作成過程がどのようにされているかという本質にプロジェクトの目的が確実に沿うようにすることが重要です。

希望する最大限に可能な解像度でビデオをオーサリングするか、そうでなければ縮小することが最善の方法です。Scaleform Video Encoder はビデオのサイズ変更を行いますが、品質の低下、伸縮、その他の問題が起きる可能性があります。低い解像度でオーサリングされたビデオを拡大しようとするときに、これは特に当てはまります。高解像度でビデオをオーサリングして、その後、特定のプラットフォームに必要な低い解像度に下げる方法のほうがはるかに優れています。

ほとんどのビデオ編集プログラムは、Scaleform Video と組み合わせて使用することができます。最善の結果を得るには、ビデオは圧縮していない AVI ファイルでなければなりません。Scaleform Video ファイルをエンコードするとき、圧縮していないファイルを使うことが不可欠です。AVI がビ

デオを圧縮する (その結果、アーティファクトの原因となる) 別のコーデック (DIVX など) で作成された場合、そのビデオを Scaleform Video フォーマットに変換すると二重圧縮となり、ビデオのサイズは小さくならない上にビデオの品質に大きく否定的な影響を与えます。

ビデオをオーサリングするとき、オーディオについて考慮することは重要です。モノラル、ステレオ、5.1 のサポートが必要かどうかを知ることは重要ですが、オーディオのローカライズはもっと重要かもしれません。吹き替えファイルをどのように作成するかを考えることは重要です。特に、(ただ字幕を付けるだけでなく)、言語ごとに吹き替えをすべて翻訳してレコーディングし直すことを計画するときはそうです。Scaleform Video は複数のセンター チャンネル オーディオ トラックを有効にしています。これを使ってさまざまな言語の吹き替えを楽に再生することができます。各言語に個別のビデオを使う代わりに、1 つのビデオを複数の言語に使うことが可能になります。ただし、この手法を使う場合、吹き替えのオーディオをビデオ/オーディオ編集プログラムから個別にエクスポートして、実行時に組み込みできるようにする必要があります。

もう一度言いますが、ビデオの制作を始める前に、そのゲームのビデオの必要条件をすべて検討し、開発過程を通じてテストを行い、時間の浪費や長期のパフォーマンス問題を避けることをお勧めします。

7.1 アルファ チャンネル

ビデオ編集やコンピュータ グラフィックの経験がないユーザーは、アルファ チャンネルやマスクという概念にはなじみがないかもしれません。Scaleform Video はピクセルごとのアルファ チャンネルを有効にします。つまり、ビデオ内の各ピクセルは塗りつぶしや、ある程度の透明度を持つことができるということです。一般的なビデオ編集プログラムでは、ビデオは透明度を制御するチャンネルに割り当てられている色に応じて、完全に、または部分的に透明な箇所を持つことができます。

これは「グリーン バック」を使ったテレビや映画の中で最も一般的です。俳優はグリーン一色の背景 (ブルーの場合もあります) の前で撮影されます。こうするとその背景を削除して、俳優を別のシーンに楽に合成することができます。明るいグリーン、またはブルーが使用されることが最も多く、これはこの 2 色が目立ち、通常は映画の他の部分には存在しない色なので、画面上で他のものとの調和を壊すことが少ないからです。ビデオは完全に、または部分的に透明になるようにエンコードできます。

ほとんどのプログラムではアルファ チャンネルの追加は比較的簡単です。アルファ チャンネルをビデオに追加するエンコーディングは、チェックボックス ボタンをクリックしてオンにするように単純な場合があります。ただし、アルファ チャンネルを持つビデオは、アルファを持たない同じビデオよりも、レンダリングに 2 倍かかる場合があるということを覚えておいてください。ゲームでは慎重にアルファ チャンネルを使用してください。特にそのゲームがリアルタイム ゲーム エンジンで使用されている場合は、注意が必要です。

8. 技術的統合について

この章では、Scaleform Video をお使いのエンジンに統合するために必要な技術情報を提供しています。

8.1 Scaleform のサウンド システムの初期化

Scaleform サウンド システムを初期化するには、GFx::Audio クラスのインスタンスを GFx::Loader オブジェクトに設定する必要があります。その目的は、サウンド レンダラ オブジェクト SoundRenderer と、SWF ストリーミング サウンドの再生のための同期パラメータを提供することです。SoundRenderer クラスは抽象 C++インターフェイスで、これをゲームに実装して、音を作り出す必要があります。Scaleform はデフォルトのインプリメンテーションを備えています。これは FMOD クロス プラットフォーム サウンド ライブラリに基づいており、サポートされているすべてのプラットフォームで使用できます。このインプリメンテーションのインスタンスを作成するには、Sound::SoundRendererFMOD::CreateSoundRenderer メソッドを呼び出して、次に作成したオブジェクトを FMOD::System オブジェクトで初期化する必要があります。

例:

```
// サウンド レンダラ オブジェクトを初期化する
FMOD::System* pFMOD = NULL;
result = FMOD::System_Create(&pFMOD);
if (result != FMOD_OK)
      exit(1);
result = pFMOD->getVersion(&version);
if (result != FMOD_OK || version < FMOD_VERSION)
      exit(1);
result = pFMOD->init(64, FMOD_INIT_NORMAL, 0);
if (result != FMOD_OK)
      exit(1);
Ptr<SoundRenderer> psoundRenderer =
                        *SoundRendererFMOD::CreateSoundRenderer();
if (!psoundRenderer->Initialize(pFMOD))
      exit(1);

// ローダ上に GFx::Audio のインスタンスを設定する
Ptr<GFx::Audio> paudio = *new GF::xAudio(psoundRenderer);
loader.SetAudio (paudioState);
```

8.2 Scaleform のビデオ再生システムの初期化

Scaleform のビデオ再生システムを初期化するには、Video クラスのインスタンスを GFx::Loader オブジェクトに設定する必要があります。この目的は、アプリケーションがビデオ ファイルの再生を必要とするときに、ビデオ プレーヤーのインスタンスを作成することです。そのビデオ ファイルがオーディオ データを備えていて、再生する必要がある場合、VideoSoundSystem インターフェイスのインスタンスを GFx::Video::Video オブジェクトに設定する必要があります。Scaleform は、サポートされている各プラットフォームのこのインターフェイスのインプリメンテーションを備えており、1 つが SoundRenderer インターフェイスに基づいて実装されています。

例:

```
// ローダ上に Video のインスタンスを設定する オプションのパラメータでビデオをエンコード
// するスレッドの優先順位を指定することができます
Ptr<GFx::Video::Video> pvc = *new Video (Thread::NormalPriority);
// SoundRenderer インターフェイスに基づいたビデオ サウンド システム インスタンスを
// 設定する
if (psoundRenderer)
{
      pvc->SetSoundSystem(psoundRenderer);
}
// 特定のプラットフォームに特有のビデオ サウンド システム インスタンスを
// 設定する
// 注意: SetSoundSystem メソッドを Video のインスタンスにつき 1 回だけ呼び出す必要がある
 pvc->SetSoundSystem(Ptr<Video::VideoSoundSystem>(*new
                                    VideoSoundSystemDX8(0)));

loader.SetVideo(pvc);
```

Windows では、Video::Video クラスは、VideoPC と名付けられ 3 つのパラメータを使います。第一のパラメータは、ビデオをエンコードするスレッドを指定します。第二のパラメータは、ビデオエンコーディングに使われるスレッドの本数を指定し、第三のパラメータは、各デコーディングに使われるスレッドのアフィニティ・マスクの配列です。

```
VideoPC(Thread::ThreadPriority
         decodingThreadsPriority = Thread::NormalPriority,
         int decodingThreadsNumber = MAX_VIDEO_DECODING_THREADS,
            UInt32* affinityMask = NULL);
```

例：

```
UInt32 affinities[] = {
                      DEFAULT_VIDEO_DECODING_AFFINITY_MASK,
                      DEFAULT_VIDEO_DECODING_AFFINITY_MASK,
                      DEFAULT_VIDEO_DECODING_AFFINITY_MASK };

Ptr<Video::Video> pvc = *new Video::VideoPC(Thread::NormalPriority,
```

```
                                        MAX_VIDEO_DECODING_THREADS, affinities)
```

PS3 と Xbox360 プラットフォームの場合、Scaleform は追加の初期化パラメータを使用する Video クラスの特殊なバージョンを提供します。

PS3 の場合、これは GFx::Video::VideoPS3 という名前で、4 つのパラメータを使用します:

```
VideoPS3(CellSpurs* spurs, int spu_num,
               int ppu_num, Thread::ThreadPriority ppu_priority,
               int spu_num, UInt8* spurs_priorities = NULL );
```

spurs –	CellSpurs 構造のポインタ
spu_num –	ビデオのデコーディングに使用できる SPU の数
ppu_num –	ビデオのデコーディングに使用できる PPU スレッドの数
ppu_priority –	PPU スレッドの優先順位
spurs_priorities –	SPU コアの優先順位

例:

```
UInt8 spurs_priorities[] = {1,1,1,1,0,0,0,0};
Ptr<Video::Video> pvc = *new VideoPS3(GetSpurs(), 0, 1000,
                                                    4, spurs_priorities);
```

PPU 利用率の低減例：

PPU 利用率ができるだけ小さくなるように Scaleform Video を初期化することが可能です。たとえば、PPU 利用率を 5%以下に抑えながら、720p ビデオを 50～60FPS で再生することができます。このような状況は、ビデオデコーディング・スレッドをプリエンプトするにはメインスレッドが十分にビジーな状態のときに発生します。たとえばゲームの動作中は、このような条件が成立するほどメインスレッドは十分にビジーです。

デコーディング・スレッドの PPU 利用率は他のスレッドの負荷によって変わります。メインスレッドがビジーではない場合、デコーディング・スレッドは PPU のおよそ 40%を使用します。ただし、以下の設定を行った場合、メインスレッドがビジーであればデコーディング・スレッドよりも優先度が高くなり、デコーディング・スレッドはリソースの 4%ないし 5%しか使用しません。実際に優先度を適切に設定すればデコーディング・スレッドの利用率を 1%にまで下げられます。

```
UInt8 spurs_priors[] = {0,0,1,1,1,1,0,0};
Ptr<Video::Video> pVideo = *new VideoPS3(GetSpurs(), 0,
                          Thread::BelowNormalPriority, 4, spurs_priors);
```

Xbox360 の場合、このステートは、VideoXbox360 と呼ばれ、ハードウェアのどのスレッドをビデオデコーディングに使うかを指定する三つのパラメータを持ちます。第一のパラメータは、内部的に最も多くの仕事をする二本のスレッドを指定します。そこで、この二本のスレッドが重複しないように

します。第二のパラメータは、ビデオデコーディングを行うスレッドの優先順位を指定します。第三のパラメータは、ビデオ読み込みに使われるハードウェア上の追加スレッドを指定します。

```
VideoXbox360(unsigned deCodingProcs = Xbox360_Proc1 | Xbox360_Proc3 |
                        Xbox360_Proc5, Thread::ThreadPriority
                        decodingThreadsPriority = Thread::NormalPriority,
                        Xbox360Proc readerProc = Xbox360_Proc2);
```

例:

```
Ptr<Video::Video> pvc = *new VideoXbox360(Xbox360_Proc1 |
                                          Xbox360_Proc3|Xbox360_Proc5,
                                          Xbox360_Proc2);
```

8.3 バックグラウンドゲームデータローディング API

ビデオプレイバック中に行われるゲームデータのバックグラウンドロード処理は、コールバックインタフェース Video::VideoBase::ReadCallback と、メソッド VideoBase::IsIORequired および VideoBase::EnableIO によって実装されています。

ReadCallback は、ビデオ読み出し動作が必要なとき、および読み出し動作が完了したときに、アプリケーションに通知するコールバックインタフェースです。

例：

```
// Simple read callback implementation for the video system
class VideoReadCallback : public VideoBase::ReadCallback
{
      public:
      VideoReadCallback() {}
      virtual ~VideoReadCallback() {}

      virtual void OnReadRequested()
      {
          // Video read started
          VideoReadStart = Timer::GetTicks();
          // ...
      }
      virtual void OnReadCompleted()
      {
          // Video read completed
          GameReadStart = Timer::GetTicks();
          // ...
```

```
        }

        UInt64  VideoReadStart;
        UInt64  GameReadStart;
    };

    // Init video system
    Ptr<Video::Video> pvc = *new Video::VideoPC(
                Thread::NormalPriority, MAX_VIDEO_DECODING_THREADS, affnts);

    // Create and set video read callback
    Ptr<VideoReadCallback> pvideoReadCallback = *new VideoReadCallback;
    pvc->SetReadCallback(pvideoReadCallback);
```

IsIORequired はビデオシステムが任意のビデオデータリード動作の実行を必要としているかを判断します。このメソッドが true を返したときは、アプリケーションは速やかにすべてのディスク IO 動作を停止し、*EnableIO* をコールしてビデオ読み出し動作を有効にしなければなりません。

EnableIO は、ゲームデータのバックグラウンドローディングに必要なビデオ読み出し動作を、有効または無効に設定します。

例：

```
    if(pvc->IsIORequired())
    {
        pDataLoader->Enable(false);
        pvc->EnableIO(true);
        fprintf(stderr, "Video is loading\n");
    }

    // ...

    if(!pvc->IsIORequired())
    {
        pvc->EnableIO(false);
        pDataLoader->Enable(true);
        fprintf(stderr, "Game data is loading\n");
    }
```

ビデオバックグラウンドロード機能のデモとして、Apps\Samples\Video ディレクトリに全ソースコードを収録した 2 つのデモ（VideoBGLoadFlash と VideoBGLoadData）を用意しています。

- VideoBGLoadFlash - ビデオの再生中に、開発アプリケーションに Flash コンテンツをロードする、単純な実装を示したデモです。
- VideoBGLoadData – ビデオの再生中に、開発アプリケーションに任意のデータをロードする方法を示したデモです。

8.4 Scaleform VideoSoundSystem インタフェース

GFx::Video::VideoSoundSystem は Video::Video プレイバックでサウンドをサポートするアブストラクトインタフェースです。デベロッパ側でこのクラスのカスタム版を作成すれば、ビデオサウンドの実装を代替することができます。ビデオをプレイする前には、このクラスのインスタンスの作成と Video::SetSoundSystem()へのインストールが必要です。一般に、このインタフェースを実装したくない場合は、プラットフォーム固有の実装を用います。

Video ディストリビューションに含まれるサウンドシステムインタフェース：

- Video::VideoSoundSystemDX8 – Windows の DirectSound
- Video::VideoSoundSystemXA2 – Windows および Xbox 360 の XAudio2
- Video::VideoSoundSystemPS3 – PS3 の MultiStream
- Video::VideoSoundSystemWii – Wii サウンドシステム
- Video::VideoSoundSystemFMOD – FMOD にもとづくサウンドインタフェース
- Video::VideoSoundSystemWwise – Wwise にもとづくサウンドインタフェース

Scaleform の現在のバージョンでは、ビデオサウンドサポートは埋め込み Flash サウンドプレイバックとは切り離されていて、汎用サウンドエンジンを必要とせずにビデオが使用できるようになっています。この動作を実現するために、Scaleform の他の部分で使われている Sound::SoundRenderer（8.1 を参照）とは切り離された、独立した Video::VideoSoundSystem クラスによって、ビデオをサポートしています。このことは、ビデオサウンドサポートが必要な場合は、SoundRenderer よりもはるかに単純な VideoSoundSystem クラスと VideoSound クラスのみを実装すればいいことになります。すでに SoundRenderer 実装をしている場合、SoundRenderer は機能のスーパーセットを備えているため、Video の初期化に SoundRenderer を直接使用してもかまいません。場合によっては 2 つの実装を混在することもできます（汎用サウンドエンジンよりもカスタムビデオサウンドクラスの方がよりよいストリーミングサポートをしている場合に有用）。

例：

```
#include "Video/Video_VideoSoundSystemXA2.h"
pvc->SetSoundSystem(Ptr<Video::VideoSoundSystem>(*new VideoSoundSystemXA2(0, 0)));
```

ビデオサウンドサポートでは GFx::Video::VideoSoundSystem と GFx::Video::VideoSound を実装する必要があります。一般にビデオ初期化中は、VideoSoundSystem のひとつのインスタンスしかインストールされません。VideoSoundSystem は、個々のビデオサウンドストリームを表す VideoSound オブジェクトの生成に使われるシングルメソッド Create を提示します。新しいビデオが開くたびに Scaleform はこの機能をコールします（同時に複数のビデオ再生が可能）。各 VideoSound オブジェクトが生成されたあと、オーディオ出力の Start と Stop を指示するために、Scaleform は複数の関数をコールします。ストリームの実際のサウンドデータは VideoSound::PCMStream のポーリングを通じて取得します。ポーリングは一般に、有効なサウンドをサービスするために、VideoSoundSystem によって維持される別のスレッドで実行されます。

Audio Input プラグインで作られた VideoSoundSystemWwise の現在の実装は、v2009.2.1 build 3271 以来、Wwise SDK の一部として提供されています。Audiokinetic 社が提供する全ソースコードと、このプラグインの Visual Studio ソリューション／プロジェクトは、*SDK\samples\Plugins\AkAudioInput* に格納されています。詳細は Audiokinetic の Wwise ドキュメントを参照してください。なお、GFx::Video::Video ディストリビューションには Wwise SDK のいかなる部分も含まれていません。個別にインストールする必要があります。

例：

```
#include "Video\Video_VideoSoundSystemWwise.h"
Ptr<Video::VideoSoundSystem> wwise =
                    Ptr<Video::VideoSoundSystem>(*new VideoSoundSystemWwise());
pvc->SetSoundSystem(wwise);
wwise->Update();
```

Apps\Samples\FxPlayer および *Apps\Samples\Common* ディレクトリにある Scaleform Player（FxPlayerAppBase.cpp、FxSoundFMOD.cpp、FxSoundWwise.cpp など）のソースコードを参照してください。また、サウンドシステムインタフェース（Video_VideoSoundSystem*.cpp）の実装については *Src\Video* ディレクトリを参照してください。

8.5 多言語ビデオ

標準的な多言語ビデオは各言語を収録した複数のオーディオ トラックで構成されています。GFx::Video::Video では、ActionScript ビデオ エクステンションを使用することで、多言語ムービー データを効率的に扱えるようになっています。たとえば、英語とスペイン語の両方のステレオ オーディオを持つビデオを考えてみましょう。このビデオの再生にあたっては、いずれかの言語のオーディオ トラックを選択します。オーディオ トラックを選択するには、NetStream から *audioTrack* エクステンションを使用します。
例:`ns.audioTrack = 2;`

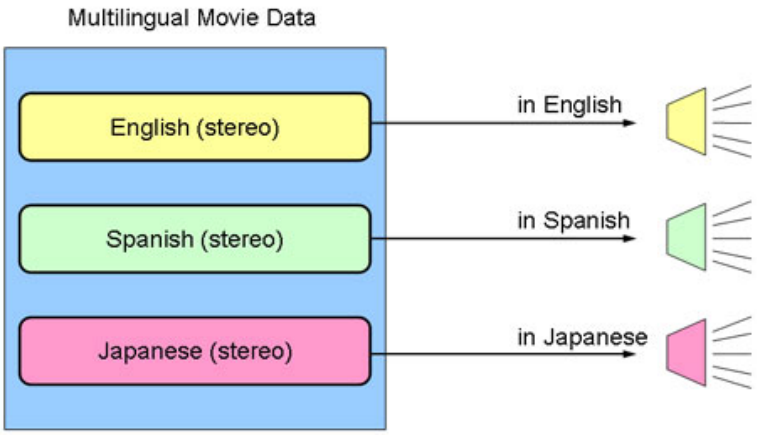

Figure 12: 多言語ビデオ

多言語サラウンド サウンド ビデオとは、共通的な 5.1 オーディオ トラックと複数の言語トラックで構成されたビデオを指します。言語トラックはサラウンド サウンドと一緒に切り替えることができますが、多言語トラックとの組み合わせではひとつのサラウンド サウンド トラックしか使えません。複数のサラウンド ビデオにおいて単一のサウンド トラックを共有することで、ムービーのサイズを抑えています。
GFx::Video::Video は次のような 2 種類の再生機能を有します。

1. センター チャネルの置き換え
2. SubAudio の再生

8.5.1 センター チャネルの置き換え

上述のとおり、多言語サラウンド サウンド ビデオではムービー ファイル サイズを抑えることで、多くの言語をサポートできるようにしています。多言語サラウンド サウンド ビデオは、共通的な 5.1 オーディオ トラックと複数の言語トラックで構成されたビデオです。通常、5.1 チャネル オーディオのオリジナルのセンター トラックの代わりにモノラル音声トラックが再生されます。多言語サラウンド サウンド ビデオを再生する場合、5.1 オーディオ トラックとモノラル音声トラックをそれぞれ個別に選択する必要があります。
ミュージックまたはサウンド エフェクトをセンター チャネルから再生する場合、これらを前もって音声トラックとミキシングしておかなければなりません。
例:

```
ns.audioTrack = 0;      // main 5.1 audio track
ns.voiceTrack = 5;      // mono audio track
```

メイン トラックが 5.1 オーディオではない場合、または音声トラックがモノラルの場合、音声トラックの設定は無視されます。メイン オーディオ トラックが音声トラックとして設定されている場合はセンター チャネルは通常どおり再生されます。

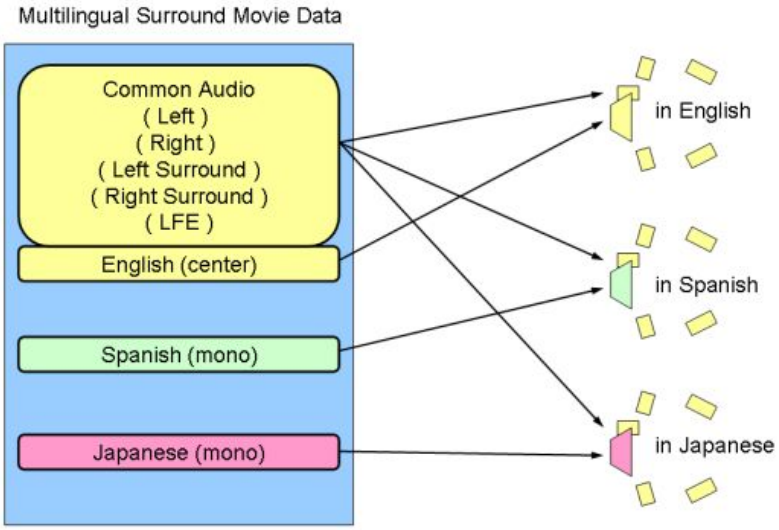

Figure 13: センターチャネルの置き換え

8.5.2 SubAudio 再生

SubAudio 再生機能を使用すると、アプリケーションは GFx::Video::Video を通じて、他のトラックとメイン トラックとを同時に再生することができます。正確には GFx::Video::Video は、メイン トラックとともにサウンド データを追加サウンド インタフェースを通じて出力します。つまり、SubAudio トラックを音声トラックとして出力することで、アプリケーションは多言語サラウンド サウンド再生を実行することができるのです。

例:

```
ns.audioTrack = 0;      // main audio track
ns.subAudioTrack = 6;   // secondary audio track
```

サブオーディオプレイバックにご注目ください。これにはメーンオーディオトラックとサブオーディオトラックのための独立したボリュームコントロール機能があります。 Sound.setVolume()の拡張されたシンタックスが使用されています。

例:

```
var audio:Sound = new Sound(this);
audio.setVolume(80, 50);       // set volume for main and subaudio tracks
```

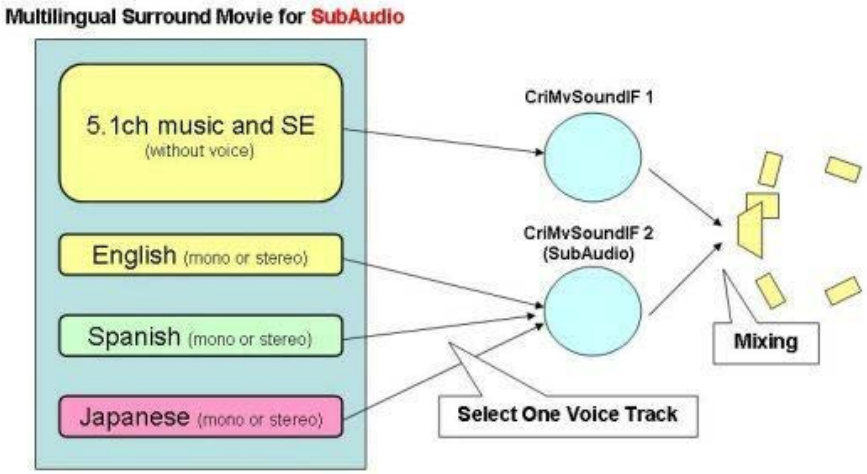

Figure 13: SubAudio 再生

9. トラブルシューティング

ビデオのコンテンツがスムーズに再生されず、音が途切れる場合、ハードウェアに十分なリソースがなくビデオのデータを十分に処理できないことが一般に考えられます。これは次のように改善できることが良くあります。

1.コードでリソースを解放する（つまり、ビデオ再生と同時に負荷の多いプロセスや IO 負荷がかかっていないようにする）。

2.ビデオデータのサイズとビットレートを減らす。

3.ビデオシステムにより多くのリソース（スレッド、フレームプール）を割り当てる。

具体的には、ビデオをスムーズに再生するには(優先順位の高い順に）次の 3 つの事項があります。

1. スレッドの設定 – 数と優先順位の変更。どのスレッドがゲームエンジンですでに使用され、または使用されていないかを考えてください。高解像度のビデオ（720P 以上）は通常 3～4 つのスレッドを要します。スレッド設定に関する詳細は先述のセクション 8 を参照してください。

2. ビデオの解像度とビットレート – ビデオムービーのサイズとビットレートのどちらか、あるいは両方を減らして負荷を軽減する必要がある可能性があります。Scaleform は必要に応じて自動的に解像度の低いビデオをスケールアップしてビューポートに合うようにします。

3. バッファーとフレームプール - ビデオ再生の設定には多くの ActionScript 方法とエクステンションがあります。次の 2 つの呼び出しはビデオシステムの使用するメモリーとバッファ時間の調節に使用します。詳細は先述のセクション 6 を参照してください。）
   a. `NetStream.setNumberOfFramePools(numPools:Number)`
   b. `NetStream.setBufferTime(bufferTime:Number)`