



# Flash Lite 2.0을 위한 콘텐츠 최적화

Josh Ulm  
수석 디자이너  
사용자 경험 개발, 모바일 및 디바이스

도움 주신 분:  
Rosalind Morrison(Adobe 컨설팅), Walter Luh(Flash Lite 엔지니어링), Jian Zheng(Flash Lite 엔지니어링),  
Jeremy Clark(XD)

# 목차

소개 .....	3
알려진 문제점 .....	4
프레임 속도 .....	4
주의 사항 .....	5
스테이지 크기 및 제작 프로파일 .....	5
Intervals 및 리스너 .....	5
프레임 건너뛰기 .....	5
전역 변수 및 로컬 변수 .....	5
명확하게 정의되지 않은 함수 .....	5
비트맵 크기 조절 및 회전 .....	5
그래디언트 줄무늬 .....	5
컨텐츠 최적화 .....	6
아트웍 포맷 .....	6
압축된 JPG .....	6
투명 PNG .....	6
그래디언트 .....	6
복잡하게 구성된 벡터 .....	6
벡터 모서리와 곡선 .....	6
모양 외곽선 .....	6
임베드된 텍스트 .....	6
여러 행으로 구성된 동적 텍스트 .....	6
무비 클립 숨기기 .....	7
트윈 영역 .....	7
레이어 유형 .....	7
간결한 소스 파일 .....	7
첫 번째 프레임 초기화 .....	7
Math 함수와 부동 소수점 숫자 .....	7
Math 루틴 데이터 .....	7
루프 반복 .....	7
XML 데이터 .....	8
저자 소개 .....	9

# 소개

본 문서에서는 Flash Lite 2.0이 탑재된 모바일 디바이스에서 사용할 파일 크기가 작고 빠르게 작동하는 Flash 콘텐츠를 제작하는 데 유용한 여러 가지 팁과 정보를 제공합니다.

대부분의 Flash 개발자는 Flash 콘텐츠를 최적화하는 데 필요한 일반적인 규칙(복잡하게 구성된 아트웍에 대한 애니메이션 사용 금지, 한 번에 많은 객체의 트윈 효과 적용 금지, 투명도 과다 사용 금지 등)에 대해 잘 알고 있을 것입니다. 데스크탑에서 Flash 8을 사용할 때 이와 같은 성능 문제는 모두 해결되었습니다. 그렇지만 Flash Lite 개발자라면 아직도 성능 문제에서 완전히 자유로워졌다고 할 수 없습니다. 대부분의 모바일 디바이스에서 제공하는 성능을 고려해 볼 때 화려하고 멋진 효과는 배제되어야 합니다. 더구나 제품별로 성능에 차이가 있고 상황에 따라서는 그 차이가 대단히 크게 나타납니다. 그리고 모바일 저작의 경우 매우 다양한 디바이스를 대상으로 퍼블리싱하게 되므로 모바일 콘텐츠를 저작할 때는 가장 낮은 표준 사양을 기준으로 작업해야 합니다. 쉽게 말해 여러 섹션에서 1바이트씩만 줄여도 최종적인 성능에는 큰 차이가 날 수 있습니다.

개발자들은 일반적인 저작 기술 중 일부를 사용하지 말라는 권장 사항을 자주 접하게 될 것입니다. 그렇지만 벡터, 동적 텍스트, 애니메이션을 사용하지 않은 Flash는 더 이상 Flash라고 할 수 없을 것입니다. 이런 기능을 전혀 사용하면 안 되는 것으로 속단하고 실망할 필요는 없습니다. 분명히 사용해야 하고 사용할 수 있지만 어느 정도 사용하는 것이 적당한 것인지를 알기 위해서는 실험을 통해서 직접 경험해 보아야 합니다. 본 문서를 읽으면서 잊지 말아야 할 것은 모바일 콘텐츠의 최적화 과정에는 대부분의 경우 성능과 단순화 작업의 상충되는 측면을 조정하는 작업이 필요하다는 것입니다. "A" 기법이 모습은 더 좋아 보일 수 있지만 "B" 기법이 성능면에서는 훨씬 탁월할 수 있습니다. 모바일 콘텐츠를 최적화하기 위해서는 콘텐츠를 배포할 디바이스에 대한 철저한 테스트 과정을 거쳐야 합니다. 실제 하드웨어에서 특정 콘텐츠 파일을 테스트하는 방법 외에 다른 대안은 없습니다. 실질적인 성능, 색상, 텍스트 가독성, 물리적인 인터랙션, UI 응답성 그리고 궁극적으로 진정한 모바일 경험을 확인하기 위해서는 반드시 실질적인 테스트 과정을 거쳐야 합니다.

# 알려진 문제점

## 프레임 속도

호스트의 timer-resolution이 해당 디바이스의 Flash Lite에서 수행할 수 있는 최대 프레임 속도에 핵심적인 영향을 미치게 됩니다. 그러므로 저작 과정에서 지정된 fps(예상 fps)가 실제 fps라고 생각하면 안 됩니다. 심지어 콘텐츠가 없는 빈 SWF 파일(렌더링 또는 AS 없음)의 경우에도 마찬가지입니다. 제작 중인 콘텐츠에서 필요한 실질적인 최대 fps를 파악하려면 예상 fps를 고려하여 조정해야 합니다.

프레임 속도를 Flash 콘텐츠를 배포할 대상 디바이스의 기본 fps로 설정합니다. 대상 디바이스의 이상적인 프레임 속도를 잘 모르는 경우 여러 가지 프레임 속도를 테스트하여, fps를 조정하면 무비의 실행 가능 프레임 속도에 크게 영향을 미치는지 여부를 확인해야 합니다.

이해를 돕기 위해 예를 하나 들겠습니다. Symbian Series 60v2 기반 휴대폰의 경우 timer-resolution이 1/64초입니다. 다시 말해서 DoPlay가 1/64, 2/64, 3/64, 4/64, 5/64 등과 같은 시간 간격으로 호출됩니다. 그렇기 때문에 해당 디바이스에서 사용할 수 있는 프레임 속도(fps)는 64, 32, 21.3, 16, 12.8 등입니다.

20fps로 실행되도록 저작된 콘텐츠는 이론적으로 1/20초에 한 번씩 호출됩니다. 그러나 Symbian에는 1/64초 timer-resolution만 있기 때문에 1/20은 Symbian에서 제공하는 3/64과 4/64 시간 간격 사이에 오게 됩니다. 플레이어는 원칙에 따라 작동되므로 1/20 시간 간격이 초과되었을 때만 프레임을 재생합니다. 예를 들면 4/64초가 지난 후에 재생됩니다. 이것은 16fps와 상응합니다. 다시 말해 예상 성능에서 20%의 조정 또는 20%의 손실이 발생한다는 것입니다.

# 주의 사항

## 스태이지 크기 및 제작 프로파일

새 무비를 생성할 때 문서가 정확하게 설정되었는지 확인하십시오. Flash 무비는 자연스러운 크기 조절이 가능하지만 원래의 기본 스테이지 크기에서 실행되지 않거나 플레이어에서 크기를 조정해야 하는 경우 성능 저하 문제가 있습니다. 문서의 스테이지 크기가 대상 디바이스의 해상도에 맞게 설정되었는지 확인하십시오. 또한 Flash Player가 "제작 설정"에서 올바른 Flash Lite 버전으로 설정되었는지, Device Central에서 적절한 디바이스 프로파일을 선택했는지를 확인하십시오.

## Intervals 및 리스너

SWF 데이터 메모리는 해당 무비 클립이 언로드되었을 때 해당 SWF 데이터를 참조하는 ActionScript 함수가 있는 경우에 가비지 컬렉션이 되지 않습니다. Intervals와 리스너를 사용했을 경우가 대표적인 예로 Intervals와 리스너가 먼저 제거되지 않으면 사용되었던 데이터는 해제되지 않습니다. `unloadMovie` 또는 `removeMovieClip`을 사용하여 콘텐츠를 제거하기 전에 `clearInterval`을 사용하여 작동 중인 intervals를 삭제하고 `removeListener`를 사용하여 작동 중인 리스너를 삭제하십시오.

## 프레임 건너뛰기

`gotoAndPlay`를 사용할 때는 현재 프레임과 요청된 프레임 사이에 있는 모든 프레임을 요청된 프레임이 재생되기 전에 모두 초기화해야 한다는 것을 기억하십시오. 두 프레임 사이에 있는 모든 프레임에 서로 다른 콘텐츠가 포함되어 있는 경우에는 타임라인보다는 다른 무비 클립을 사용하는 것이 보다 효율적입니다.

## 전역 변수 및 로컬 변수

함수에서 로컬 변수는 플레이어에서 무비가 재생될 때 전역 변수보다 훨씬 빠르게 등록됩니다. 가능하다면 `var`를 사용하십시오.

## 명확하게 정의되지 않은 함수

명확하게 정의되지 않은 구문을 사용하여 함수를 정의하지 마십시오(예: `myObj.eventName = function ( ) { ... };`). 명확하게 정의된 함수가 더 효율적입니다(예: `function myFunc ( ) { ... }; my Obj.eventName = myFunc;`).

## 비트맵 크기 조절 및 회전

Flash Lite 플레이어는 비트맵 다듬기 기능을 지원하지 않습니다. 다시 말해서 사용자가 비트맵의 크기를 조절하거나 회전시키면 비트맵의 가장자리가 매끄럽지 않게 표시되거나 품질이 떨어질 수 있습니다. 비트맵을 사용할 때는 원래 크기와 회전되어 있는 그대로 사용하는 것이 가장 좋습니다. 그래픽의 크기를 조절하거나 회전시켜야 하는 경우에는 비트맵 대신에 벡터 그래픽을 사용하는 것이 좋습니다.

## 그래디언트 줄무늬

현재 시중에 출시된 대부분의 디바이스는 16비트 색상(수 천 색상)만 지원합니다(24비트 또는 32비트(수 백만 색상)는 지원하지 않음). 그러므로 그래디언트에서 색상이 자연스럽게 변화하지 않고 여러 가지 색상의 띠로 표현되는 경우가 많습니다. 이 문제를 해결할 수 있는 한 가지 방법은 타사 무료 Photoshop 필터(Telegraphics의 5\_6\_5 필터)를 사용하여 비트맵을 후처리하는 것입니다. 이 필터는 비트맵의 색상 수를 16비트로 줄여서 디더링합니다. 이 필터는 <http://www.telegraphics.com.au/sw/> 사이트에서 다운로드할 수 있습니다.

# 컨텐츠 최적화

## 아트웍 포맷

가능하면 아트웍에 벡터 포맷 대신에 비트맵 포맷을 사용하십시오. 여러 개의 벡터를 렌더링하면 성능이 저하되는 반면, 비트맵은 훨씬 빠르게 렌더링됩니다.

## 압축된 JPG

JPG의 압축 해제 성능을 저하시키는 요인이 됩니다. 그러므로 메모리가 충분하다면 PNG를 사용해 보십시오.

## 투명 PNG

PNG 파일에서 투명도 양을 최대한 줄이십시오. 플레이어는 비트맵의 투명 부분을 표시할 때 다시 그릴 영역을 계산하게 됩니다. 예를 들어 전경 요소를 나타내는 투명 PNG를 사용하는 경우 전체 크기의 화면으로 투명 PNG를 내보내지 마십시오. 전경 요소의 실제 크기로 내보내십시오.

## 그래디언트

가능한 한 벡터 그래디언트의 사용을 자제하십시오. 플레이어는 벡터 그래디언트를 계산하고 렌더링하기 위해 많은 리소스를 소모합니다. 벡터 그래디언트를 반드시 사용해야 하는 경우에는 방사형 그래디언트 대신 비교적 빠르게 렌더링되는 선형 그래디언트를 사용하십시오.

## 복잡하게 구성된 벡터

반드시 벡터 모양을 사용해야 하는 경우 가능한 최대도 최적화하십시오. 모양이 복잡하면 복잡할수록 Flash는 해당 모양을 렌더링하기 위해 더 많은 작업을 수행해야 합니다. 최적화는 아이콘 등과 같이 작은 벡터 모양에 특히 유용합니다. 아이콘은 크기가 작기 때문에 세부적인 모양이 손실되어도 문제가 없지만 만약 작은 크기에도 불구하고 복잡한 모양을 그대로 유지하려면 플레이어는 렌더링 시 더 많은 작업을 해야 합니다. 대부분의 경우 벡터 대신 비트맵을 사용하는 것이 보다 나은 방법입니다.

## 벡터 모서리와 곡선

모서리가 곡선보다 수학적으로 렌더링이 더 간단합니다. 특히 매우 작은 벡터 모양에서는 가능하면 평평한 가장자리를 사용하십시오.

## 모양 외곽선

채우기는 외부 모양만 렌더링하면 되지만 외곽선은 내/외부를 모두 렌더링해야 합니다. 다시 말해서 채우기와 선을 그리기 위한 작업량이 두 배가 된다는 것입니다. 가능하면 외곽선을 사용하지 마십시오.

## 임베드된 텍스트

텍스트는 기본적으로 매우 복잡한 벡터 모양입니다. 그렇기 때문에 Flash에서 렌더링해야 하는 매우 복잡한 유형 중 하나입니다. 일반적으로 텍스트는 반드시 필요한 요소이기 때문에 완전히 사용하지 않을 수는 없습니다. 텍스트를 사용해야 하는 경우 텍스트를 애니메이션 처리하거나 애니메이션 위에 배치하는 것을 피하고 텍스트를 비트맵으로 사용하십시오.

## 여러 행으로 구성된 동적 텍스트

줄바꿈 작업은 플레이어에서 시간이 많이 소요되는 프로세스입니다. 정적 텍스트 필드는 줄바꿈 작업이 컴파일 시에 미리 계산되기 때문에 문제가 되지 않습니다. 그렇지만 여러 행으로 된 동적 텍스트 및 입력 텍스트의 경우에는 텍스트 문자열의 줄바꿈이 캐시되지 않습니다. 줄바꿈 작업은 플레이어에서 런타임 시에 수행되며 텍스트 필드를 다시 그려야 할 때마다 매번 다시 계산됩니다. 동적 컨텐츠의 경우 동적 텍스트 필드를 사용해야 하지만 가능하다면 정적 텍스트 필드를 사용하는 것이 좋습니다.

## 무비 클립 숨기기

화면에 나타나는 무비 클립을 숨길 때 `_alpha = 0` 및 `_visible = false`를 사용하지 마십시오. 무비 클립을 보이지 않게 하거나 무비 클립의 알파를 0으로 변경하면 해당 무비 클립은 플레이어의 스캔라인 렌더링 계산에 그대로 포함되므로 성능에 영향을 미치게 됩니다. 비슷한 경우로, 무비 클립을 다른 아트웍으로 덮어 숨기지 마십시오. 그렇게 하더라도 해당 무비 클립은 플레이어의 드로잉 계산에 그대로 포함됩니다. 무비 클립을 스테이지 밖으로 완전히 이동하거나 `removeMovieClip`을 사용하여 제거하십시오.

## 트윈 영역

Flash는 애니메이션 처리된 영역을 그릴 때 해당 영역 주위에 직사각형 경계 상자를 정의하여 그립니다. 이 때 직사각형을 최대한 작게 함으로써 드로잉을 최적화할 수 있습니다. 즉, 가능한 경우 트윈을 겹치지 말아야 합니다. 왜냐하면 Flash는 겹쳐진 영역을 하나의 직사각형으로 인식하여 전체 영역이 필요 이상으로 커지게 되기 때문입니다. 플레이어에서 Flash 8의 '다시 그리기 영역 표시' 기능을 사용하여 애니메이션을 최적화하십시오.

## 레이어 유형

가능하면 비트맵 레이어는 비트맵 레이어끼리, 벡터 레이어는 벡터 레이어끼리 서로 가깝게 배치하십시오. 그 이유는 플레이어가 무비를 렌더링할 때 콘텐츠의 유형에 따라 서로 다른 렌더러를 사용하여 무비를 렌더링하기 때문입니다. 이렇게 렌더러 간을 전환하는 데에는 시간이 소요됩니다. 전환 시간이 많이 걸리지는 않지만 전환이 자주 일어난다면 분명히 성능에 좋지 않은 영향을 주게 됩니다. 벡터 콘텐츠를 포함하는 레이어를 서로 가깝게 배치하고 비트맵에 대해서도 동일하게 서로 가깝게 배치하면 플레이어는 두 렌더러 간의 전환 작업을 줄임으로써 보다 빠르게 렌더링할 수 있습니다.

## 간결한 소스 파일

두말할 나위 없이 무비의 크기는 가능한 작게 유지해야 하며 컴파일하기 전에 필요 없는 콘텐츠와 코드를 제거하십시오. 사용하지 않는 무비 클립과 불필요한 프레임 및 코드 루프는 반드시 제거하고 너무 많은 프레임이나 불필요한 프레임은 사용하지 마십시오. 믿지 않을 수도 있지만 이러한 빈 프레임이 성능 저하의 큰 요인이 됩니다.

## 첫 번째 프레임 초기화

모든 콘텐츠를 무비의 시작 부분에 배치하여 미리 로딩하는 것은 데스크탑에서는 괜찮지만 모바일 디바이스에서 이렇게 하면 결과적으로 무비가 매우 느리게 시작됩니다. 무비 클립이 필요할 때 초기화될 수 있도록 무비 전체에 걸쳐 콘텐츠를 일정한 간격으로 자연스럽게 배치하십시오.

## Math 함수와 부동 소수점 숫자

Math 함수와 부동 소수점 숫자의 사용을 최대한 줄이십시오. 이런 함수를 계산하게 되면 콘텐츠의 실행이 느려집니다.

## Math 루틴 데이터

Math 루틴을 사용해야만 하는 경우 값을 미리 계산하여 변수 배열에 저장하는 것이 좋습니다. 데이터 테이블에서 해당 값을 가져오는 방법이 런타임 시 플레이어에서 계산하는 것보다 훨씬 빠릅니다.

## 루프 반복

`for` 루프를 실행하면 반복할 때마다 조건을 확인하게 되고 이때 오버헤드가 발생하여 성능에 나쁜 영향을 주게 됩니다. 반복에 따른 리소스 소모와 루프 오버헤드가 비슷한 경우에는 여러 개의 작업을 개별적으로 실행하도록 루프를 사용하십시오. 이렇게 하면 코드의 크기는 커지지만 보다 빠른 성능을 얻을 수 있습니다.

## XML 데이터

가능하면 XML 파일을 로딩하거나 파싱하지 마십시오. 이 작업은 프로세서 리소스를 소모하여 성능에 나쁜 영향을 미칩니다. 가능하면 데이터를 간단한 이름/값 쌍으로 저장하고 loadVars를 사용하여 텍스트 파일에서 로딩하거나 미리 컴파일된 SWF 파일에서 로딩하십시오.

## 저자 소개

Josh Ulm은 Adobe Systems, Inc의 모바일 및 디바이스 사업부에서 사용자 경험 개발 수석 디자이너로 근무하고 있습니다. 2004년 Macromedia에 입사한 이후 Adobe로 합병되어 지금까지 모바일 및 디바이스 사업부에서 모바일 경험 플랫폼을 정의하는 작업에 매진해 왔으며 지금도 더 멋진 Flash Lite 경험을 개발하기 위해 개발자 및 고객과 긴밀히 협력하고 있습니다. 그는 회사와 고객을 위해 많은 제품과 기술을 성공적으로 채택할 수 있도록 하는데 일조했으며, Adobe의 경험에 대한 비전을 제시하고 개발하는 임무를 맡고 있으면서 Flash 및 모바일 개발자 커뮤니티에서 활동적이고 존경 받는 전문가로 정평이 나 있습니다.