



最佳化 Flash Lite 2.0 內容

Josh Ulm
首席設計工程師
用戶體驗設計主管，行動與裝置

貢獻者：
Rosalind Morrison，Adobe 諮詢；Walter Luh，Flash Lite 工程；Jian Zheng，Flash Lite 工程；Jeremy Clark，XD

目錄

簡介	3
已知問題	4
影格速率	4
牢記事項	5
舞台尺寸和發佈設定檔	5
間隔與聆聽器	5
跳過影格	5
全域和本端變數	5
匿名函數	5
縮放和旋轉點陣圖	5
漸層跳階 (Gradient banding)	5
最佳化內容	6
作品格式	6
壓縮 JPG	6
透明 PNG	6
漸層	6
向量複雜性	6
向量稜角和曲線	6
形狀外框	6
內嵌文字	6
多行動態文字	7
隱藏影片片段	7
漸變範圍	7
圖層類型	7
清理原始檔	7
初始化第一個影格	7
Math 函數和浮點數	7
Math 常式資料	7
迴圈重複	8
XML 資料	8
關於作者	9

簡介

本文件包含許多實用秘訣和見解，協助您使用 Flash Lite 2.0 可順利建立適用於行動裝置之精簡、快速的 Flash 內容。

大部分的 Flash 開發人員均十分瞭解最佳化 Flash 內容的通則，例如不要製作龐大複雜的作品動畫，不要立即建立太多物件的漸層，以及不要濫用透明功能。桌上型電腦版 Flash 8（終於！）解決了這些效能問題。但是，Flash Lite 開發人員還需要應付更複雜的狀況。我們目前沒有足夠的力量在裝置上執行好萊塢式效果（至少目前是如此）。此外，某些裝置的效能會比其他的好，有時兩者相差甚鉅。而且，由於製作行動內容通常需要發佈至許多不同的裝置，所以我們必須針對最低階的常見裝置來製作內容。也就是說，光是一個位元就可能造成很大的差異。

許多人會建議避免使用一些常用的製作技術。顯然，Flash 少了向量圖形、動態文字和動畫，就不能算是 Flash，所以不要因為這些功能有其侷限就感到失望。只是您需要嘗試出哪些可用、哪些不適用，找出這些功能的極限何在。當您閱讀本文時，請記住最佳化行動內容通常需要衡量得失。技術 "A" 看起來可能更好，但是技術 "B" 產生的效能可能更棒。在衡量得失時，您必須瞭解和預期在最佳化行動內容的過程中需要在目標裝置上進行多次測試。沒有別的測試可以替代實際在硬體上進行的檔案測試，也沒有其他方式更能確保實際效能、實際色彩、文字可識別度、實際互動、UI 回應速度以及實際的行動使用體驗。

已知問題

影格速率

主機的計時器精度會大幅影響 Flash Lite 在該裝置上達成的最大影格速率。因此，即使 SWF 是空的（無演算或 AS），在製作階段指定的 fps（預期的 fps）並不會化為實際的 fps！您必須進行調整，才能在執行內容時達到預期的最大 fps。

因此，請將影格速率設為目標裝置本身的 fps。如果您不確定您目標裝置的實際影格速率，您應測試不同的影格速率，確定調整速率是否會對影片可用的影格速率有大幅影響。

例如：在 Symbian Series 60v2 手機上，計時器精度為 1/64 秒。這表示可在下列時間間隔呼叫 DoPlay：1/64、2/64、3/64、4/64、5/64 以此類推。因此，在此類裝置上的可用影格速率 (fps) 為：64、32、21.3、16、12.8 以此類推。

當執行影格速率為 20 fps 的內容時，理論上應可每 1/20 秒呼叫一次。但是，Symbian 的時間精度為 1/64 秒，而 1/20 則介於 Symbian 可提供的 3/64 和 4/64 間隔之間。由於播放器的間隔較小，只有在超過 1/20 間隔（即 4/64 秒之後）才會播放下個影格。這相當於 16 fps！所以需要 20% 調整 — 或是損失感受效能！

牢記事項

舞台尺寸和發佈設定檔

當建立新影片時，請確定您的文件已設定正確。雖然 Flash 影片可順暢地縮放，但是如果影片未依其原本的舞台尺寸執行，而必須在播放器上進行縮放時，就會有效能問題。請確定您設定文件的舞台尺寸符合目標裝置的解析度。此外，也請確定在「發佈設定 (Publish Settings)」之下，將 Flash Player 設為正確的 Flash Lite 版本，並在 Device Central 中選擇了適合的裝置設定檔。

間隔與聆聽器

當影片片段未移除時，如果有任何 ActionScript 函數仍參照至 SWF 資料，將不會回收 SWF 資料記憶體。使用間隔與聆聽器就是其中兩例，必須先清除 ActionScript 後，才能釋放資料。請確定使用 *clearInterval* 以清除任何作用中的間隔，然後使用 *unloadMovie* 或 *removeMovieClip* 將作用中的聆聽器移除。

跳過影格

當使用 *gotoAndPlay* 時，請記住目前影格以及要求影格之間的每個影格均需要先初始化，才能播放要求的影格。如果兩影格之間的每個影格均包含不同的內容，則使用不同的影片片段會比時間軸要來得更有效率。

全域和本端變數

在函數中，和全域變數相比，本端變數的登錄方式會讓播放器更快地進行擷取和設定。所以請儘量使用 `var`。

匿名函數

避免使用匿名語法定義函數：`myObj.eventName = function() { ... };`
明確定義函數會更有效率：`function myFunc() { ... }; my Obj.eventName = myFunc;`

縮放和旋轉點陣圖

Flash Lite 播放器不支援點陣圖平面化。這表示如果您縮放或旋轉點陣圖時，粒子會變粗。因此，最好使用點陣圖的原始方向和解析度。如果您必須縮放或旋轉圖片，請考慮採用向量圖形。

漸層跳階 (Gradient banding)

目前市面上大部分的裝置仍只支援 16 位元色 (千色) 而非 24 或 32 位元 (百萬色)。這表示漸層通常會顯示為實色條紋，而非平順的漸層轉換。在處理此問題是使用 Telegraphics 的第 3 方 Photoshop 濾鏡 (稱為 5_6_5) 對您的點陣圖進行後處理，以降低點陣圖的色深降至 16 位元並進行抖色。您可在此處找到此濾鏡：<http://www.telegraphics.com.au/sw/>。

最佳化內容

作品格式

請盡量在作品中使點陣圖而非向量圖形。演算多個向量圖會降低效能，而演算點陣圖時的速度會更快。

壓縮 JPG

解壓縮 JPG 將會降低效能。因此，如果記憶體容許的話，請嘗試使用 PNG。

透明 PNG

請確定將 PNG 檔案中使用的透明部分降至最低，即使是點陣圖的透明部分，播放器也必須計算重繪。例如，如果您使用透明 PNG 當做前景元素，請勿以全畫面尺寸匯出透明 PNG。請以前景元素的實際大小匯出。

漸層

將向量漸層的使用部分降至最低。因為播放器需要耗費許多記憶體來計算和演算它們。如果必須使用它們，請注意線性漸層的演算速度會比徑向漸層更快。

向量複雜性

當您一定要使用向量形狀時，請儘可能採取最佳化運用方式。形狀的尺寸愈大，Flash 演算該形狀時的工作就愈重。最佳化對於小向量形狀（例如圖示）尤其有用。您的圖示可能會由於過小，而使許多細節部分遺失。如果要保留形狀的複雜度，播放器在演算時就需要執行額外的工作。通常使用點陣圖而非向量圖是比較明智的選擇。

向量稜角和曲線

在數學計算上，演算稜角會比曲線更簡單。請儘量使用平坦的邊緣，尤其是非常小的向量形狀。

形狀外框

對於填充圖形，只需要演算外側形狀，而外框圖形則必須演算外側和內側。這表示繪製線條的工作是繪製填充圖形工作的兩倍。所以，請儘量避免使用它們。

內嵌文字

基本上，文字是非常複雜的向量形狀。對 Flash 而言，它是一種演算較複雜的形狀。當然，文字通常是基本要素，所以很難完全避免使用。當您使用文字時，避免製作動畫文字或將它置於動畫之上，使用文字時也請考慮採用點陣圖。

多行動態文字

在播放器中斷行是十分耗時的程序。靜態文字欄位則沒有問題，因為在編譯時已預先計算斷行。但是對於多行動態和輸入文字，則不會快取文字字串的斷行。在播放器執行時期完成斷行，在每次文字欄位需要重繪時就必須重新計算。對於動態內容，則無法避免使用動態文字欄位，但是可以儘可能使用靜態文字欄位。

隱藏影片片段

避免使用 `_alpha = 0` 和 `_visible = false` 來隱藏影片片段。如果您只是將影片片段的顯示 (visibility) 屬性關閉或將它的 `alpha` 值變更為零，它仍會包含在播放器的掃描線演算計算中，因此會影響到效能。同樣地，請勿嘗試將它放在作品其他元件的後面來隱藏它。它仍會包含在播放器的計算中。請使用 `removeMovieClip`，將影片片段完全從舞台移除。

漸變範圍

當 Flash 繪製動畫範圍時，會在該範圍定義矩形邊界方塊。您可儘量將此矩形設為最小，以最佳化繪圖效能。這表示，如果可以的話，您應避免重疊漸變範圍，因為 Flash 會將合併範圍視為單一矩形，而使整個範圍變大。使用 Flash 8 在播放器中的「顯示範圍功能」，以最佳化您的動畫。

圖層類型

請儘量分別將點陣圖圖層和向量圖層靠近排在一起。當播放器演算影片時，需要建置不同的演算器，視內容類型而定。切換演算器會耗用一些時間，雖然不多，但如果經常需要切換，就會使整個時間變長。將包含向量內容的圖層排在一起，並將包含點陣圖內容的圖層排在一起，可減少切換次數，讓播放器演算速度更快。

清理原始檔

不用說，您當然會希望儘可能保持影片大小精簡，並在編譯之前先移除任何多餘的內容和程式碼。請確定不需要的影片片段已移除，刪除多餘的影格和程式碼循環，並避免使用過多或額外的影格。信不信由你，這些空白影格加起來也是相當可觀的！

初始化第一個影格

雖然在桌上型電腦上，將內容置於影片開頭以預載所有內容相當合理，但是在行動裝置上這麼做，則會使影片開始播放的速度變慢。請依序隔開影片中的內容，在要用到影片片段時再將它們初始化。

Math 函數和浮點數

請將 Math 函數和浮點數使用量降至最低。計算這些函數會降低內容的效能。

Math 常式資料

如果您必須使用 Math 常式，請考慮預先計算值並將它們儲存在變數陣列中。從資料表擷取值要比播放器在執行時計算它們要快得多。

迴圈重複

執行迴圈可能會耗用很多資源，因為檢查每個重複條件時都會耗用記憶體。當重複和迴圈耗用的記憶體相當時，請展開您的迴圈以個別執行多個運算。這會使程式碼大小變大，但是效能會提高。

XML 資料

儘可能避免載入和分析 XML 檔案；這會耗用處理器的資源並會影響到效能。可能的話，請以簡單名稱/值配對儲存資料，並使用 loadVars 文字檔案或從預先編譯的 SWF 檔案載入。

關於作者

Josh Ulm 是 Adobe Systems, Inc. 行動與裝置的首席設計工程師和設計主管。在 2004 年加入 Macromedia，之後加入 Adobe 時，他主要服務於行動與裝置部門，規劃行動使用體驗平台，並直接與開發人員和客戶合作以建立引人入勝的 Flash Lite 使用體驗。在他努力下，使公司及其客戶的許多產品和技術皆成功獲得市場接受；他時常需要簡報並規劃公司的使用經驗願景。他是 Flash 和行動開發人員社群中活躍且受人敬重的成員。