Flash Lite Tips and Tricks

Version 1.0; April 16, 2008

Introduction

This article gives some recommendations regarding performance improvements. Optimization efforts are needed in order to create a high-quality movie or application. The article covers both content optimization and ActionScript optimization.

Detailed information on optimization

Content optimization

- **Do not compress movies when publishing.** Compressing movies increases the heap use because parts of the compressed file, in addition to the decompressed file, will remain in the memory.

- **Reuse graphics for animations whenever possible**. For example, in case of a video where a phone is moving across the screen, edit the graphics and cut out the phone, and move the graphic of the phone across the screen. This way you can save up to 10 pictures per second of animation, which results in a much smaller SWF file size.

- **Use key frame animations wherever possible and if not, try to use tweens**. Tweens, although suitable for applying motion to objects while authoring, put much load on the heap. Try to use key frame animations for moving objects across the screen. If this does not work or the animation is not smooth enough, use a tween.

- **If you have to use tweens, limit the affected region as much as possible**. Flash applies a bounding box around the tween region and always redraws this box. If there are overlapping tweens, there will only be one large bounding box around both tweens. Use the "Show redraw region" of Flash 8 to limit the region as much as possible.

- **Avoid using too detailed graphics**. Remember that the screen resolution of current mobile phones is typically between 128 by 160 pixels and 240 by 320 pixels and the screen diameter is about 2 inches. This means that the level of detail that the user can actually perceive is rather limited compared to a regular computer screen.

- **Test on the actual device as early and often as possible.** There is no way to realistically emulate the limitations in processor speed, heap memory, and other functions of a mobile device on a PC yet. It may be that the movie you have created runs fine in the device central emulator of the Adobe IDE, but the performance on the actual mobile device may be poor.

- **There are differences between the version of Flash Lite that ships with Nokia devices and the developer version made available from Adobe.**

- **Decide which type of content to use for each use case.** If your movie/application should be scalable and it is not absolutely necessary to present photorealistic pictures, using vector graphics and animation is a viable option for you. This also allows you to have objects turn around and twist without making the file too big. On the other hand, if the movie does not have to be scalable and you only want to show static pictures that might be animated just by using scrolling, tweening, and color tint effects and/or you need to present detailed photorealistic information, using bitmaps might be the easiest and most effective way of doing so.

- **Find a good balance between bitmaps and vector graphics.** Bitmaps usually consume a lot more memory than vector graphics, which increases the size of the exported SWF file. Moreover, the movie will not be scalable anymore. Vector graphics and animations, on the other hand are smaller, but also result in a higher load on the processor.

- **Use different layers to separate animations from static content.** This makes exporting the SWF file more effective, as static objects are clearly marked and will be exported only once.

- **Group layers that have similar content types to be next to each other.** The Flash player renders vector graphics differently from bitmap graphics. To speed up rendering, group the vector content in your .fla file to be on adjacent layers, if possible, and do the same for bitmap graphics. This enables the rendering algorithms to run faster and do less calculation for rendering the same content.

- **Optimize vector shapes before exporting the movie.** Select the shape and then either choose Modify > Shape > Optimize from the menu tab or press CTRL+ALT+SHIFT+C. To optimize the shape even more, do as follows to remove superfluous anchor points from a shape:

  o Select the shape.

  o Scale it to 10%.

  o Scale it to 10% once more.

  o Unselect the shape (the changes are actually applied only then)

  o Select the shape again.

  o Scale it to 100%.

  o Scale it to 100%.

  o Unselect the shape.

- **Use MovieClip symbols for elements and animations that are used more than once**. Converting the animation/element into a movie clip has influence on the way the element is exported. It makes the exported SWF smaller than it would be when just reusing the element.

- **Prevent using Bitmaps with an Alpha channel whenever possible**. Using alpha channels decreases the performance of the movie playback, as transparencies have to be calculated for each frame.

- **Prevent using Alpha effects in general.** When fading an object in or out, it is more effective to apply a color tint in the color of the background and fade it using tweenings.

- **Limit the types of line styles.** Each line style has to be exported to the SWF file and this increases the size of the final movie.

- **Prefer lines over brushes.** Brushes increase the sixe of the exported SWF file. However, brush effects are usually displayed faster than line effects.

- **Limit the fonts and font styles used.** Only device fonts and basic _sans style should be used. Any other difont styles have to be vectorized when exported to the SWF file, so if you know you will not need bold or italic letters, do not use them.

- **Avoid using gradients.** Each gradient uses about 50 bytes more in the SWF file than just using a single color fill.

- **Use color tint effects instead of several symbols.** If the same object appears in your movie more than once, but in different colors, just create a single symbol and use the color tint effects to let it appear in different colors.

- **To increase the frame rate, change only what is necessary.** Flash displays small deltas between images a lot faster than big changes. It also matters where you change things and the overall area affected. It usually takes longer to change two small things in different corners of the stage than having one bigger change right in the middle of the scene.

ActionScript optimization

ActionScript optimization can also influence the performance of Flash Lite movies. Read this section very carefully and review your ActionScript code thoroughly, especially if your application uses a lot of ActionScript.

- **Simplify your ActionScript whenever possible.** Try to eliminate superfluous elements such as variables and loops. Use the Omit Trace Actions option when publishing your content to automatically remove trace() statements in the SWF file.

- **Limit loops.** Only use a loop if you cannot avoid it and keep the code executed in the loop as simple as possible. For example, initializing the data in a small array can be faster when setting the single members explicitly in code although it does not look as elegant and is less flexible. When using frame-based loops, stop them as soon as they are not needed anymore.

- **Try to avoid string processing and array use.** String operations and array access always use a lot of CPU power. If you can do something using numbers instead of strings, do so.

- **Avoid inheritance.** Try to put all the functionality you need into one class rather than inheriting some from a superclass. Every superclass adds calls and uses more memory.

- **Condense package structure.** Keep the package structure used by your classes as simple as possible to limit the size of SWF files and use memory in a more efficient way.

- **Delete objects for memory optimization.** Flash manages memory internally and clears out unused objects using a garbage collector. This happens every 60 seconds or when memory use increases by 20%. You can optimize memory use by deleting objects you do not use anymore and setting local variables to null. This helps the garbage collector decide which memory segments to release.

- **Use events sensibly.** Only watch the events you absolutely need to know about. Avoid using Object.watch and Object.unwatch. Explicitly remove event listeners from objects before deleting or nullifying them using removeListener().

- **Delete custom classes from memory.** If you load custom classes as part of an SWF file, the bytecode for those classes remains in memory even though you might already have unloaded the SWF file again. You have to delete them to mark the memory used by this code from the garbage collector. To delete these classes, use the delete operator and the name of the custom class to unload using the full package path.

Further reading

[1]   Flash Lite 2.0 Content Development Kit, Adobe Inc., available at
        http://www.adobe.com/devnet/devices/flashlite.html

[2]   Series 40 UI Style Guide, available at http://www.forum.nokia.com/