White Paper

GETTING STARTED WITH ADOBE FLASH® LITE™

Keldon Rush, Smashing Ideas Inc

About This White Paper

0

Table of contents

Who Should Read This White Paper

This white paper is intended for anyone interested in Flash Lite and the process of creating mobile applications and content with Adobe's Creative Suite 3 and is primarily geared to the Flash Lite novice / beginner.

Intended Audience

- Flash Developers including Web developers, Application developers, and Software engineers
- Flash Designers including Interactive, Web, Multimedia, UX (User Experience), and Graphic designers, Creative managers, and Art directors
- Students, Educators, and Trainers
- Mobile developers interested in Flash Lite
- Mobile UX designers interested in Flash Lite
- Mobile Operators
- Mobile Device OEMs (Original Equipment Manufacturer)

Document Aim

To give an overview of the processes, tools, and some best practices involved in developing Flash Lite applications.

Designers, Developers, and Design and Development teams which are new to mobile development and/or Flash Lite development will find this document helpful to get started creating with Flash Lite.

Flash Lite Overview: What, Why, and Where

1

What is Flash Lite?

First, what is Flash?

The term 'Flash' refers to the different integral components of the Flash Platform.

1. **Adobe Flash authoring environment & Adobe Flash Video Encoder**

   These are the authoring tools that designers and developers use to create Flash projects and Flash video respectively.

   The Adobe Flash authoring environment creates FLA (**FL**ash **A**uthoring file) and AS (**A**ction**S**cript) files.

   Source files are compiled into the runtime Flash file SWF.

   Adobe Flash CS3 Video Encoder accepts a wide variety of video source files as input and re-encodes those source files into Flash video files.

2. **'Flash Files': SWF & FLV**

   SWF is the highly optimized and compact file format that is executed by the Flash Player at runtime. SWF files can contain code to be executed as well as a number of different types of objects for presentation and interaction.

   FLV (**FL**ash **V**ideo) files contain synchronized video and audio.

3. **The Flash Player**

   The 'Flash Player' refers to the runtime engine that interprets, displays, executes, and allows interaction with SWF and FLV files. There are many different versions of the Flash Player that run across different operating systems and devices.

> **Link: SWF File Format Specification FAQ:**
> http://www.adobe.com/licensing/developer/fileformat/faq/

Ok, so what about Flash Lite?

The SWF file format was already an ideal file format for mobile devices – highly optimized and compact – perfect for slower wireless networks and memory constrained devices.

As personal computing started to spread to mobile devices an increasing demand for a Flash Player that was specifically optimized for these less robust computing devices became clear.

'Flash Lite' refers to these Flash Players that are optimized to run on mobile and other non-PC devices such as MP3 players, gaming devices, vehicle in-dash computers, and other embedded scenarios where the Flash Player makes perfect sense, but the computing capabilities of the device are limited.

The 'Lite' denotes a lighter footprint in all ways: file size, memory usage, and CPU requirements.

'Lite' also denotes slightly less capabilities although with Flash Lite 3 the gap between Flash and Flash Lite is much more narrow.

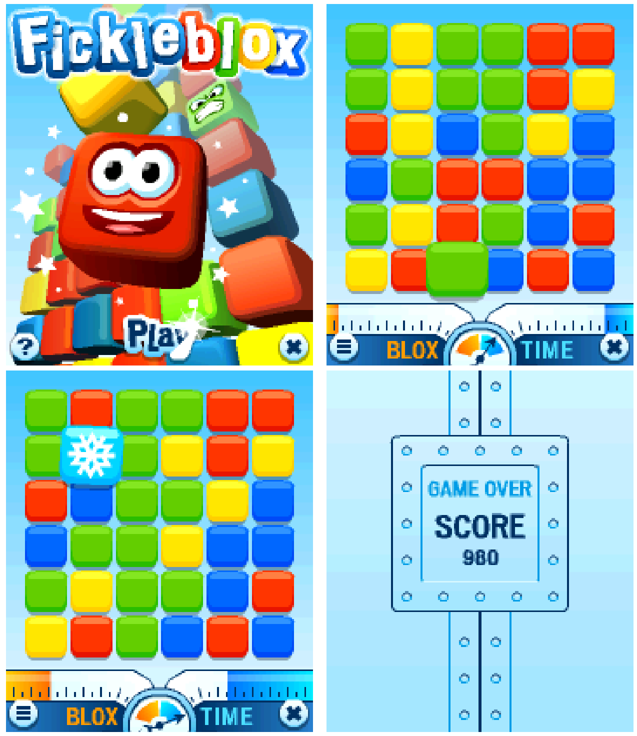
What can be created with Flash Lite?: Examples of Flash Lite UX

Flash Lite content runs the full spectrum of the mobile ecosystem. Screensavers, Wallpapers, Games, mobile marketing, and Applications as well as specific content types that only Flash can create.

Below are some examples of engaging mobile experiences created with Flash Lite.

Fickleblox

Fickle Blox is a puzzle game by Blue Sky North, a Smashing Ideas company. Users match up the colored Blox in this fast and furious game. The longer the chains, the more points a user gets. But watch out for the Fickle Blox - they have a mind of their own and can help or hinder a user along the way.

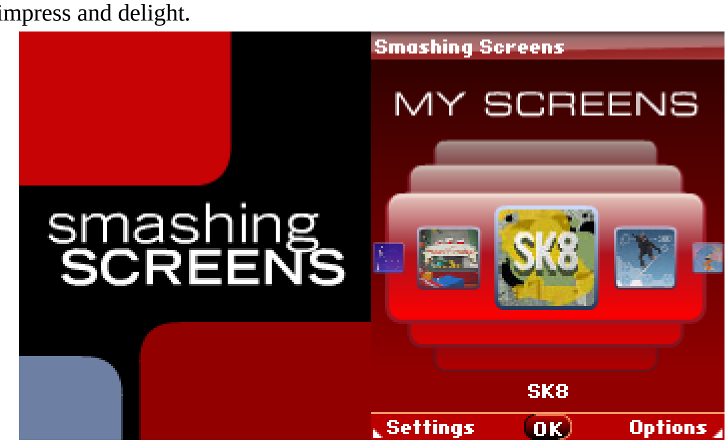


Blue Sky North
Blue Sky North, a Smashing Ideas company, designs and develops interactive applications and engaging games to Flash enabled devices. Fickle Blox is a fast and furious puzzle game that lets users build chains and gain points along the way.

Smashing Screens: Screensavers

Smashing Screens offers advanced navigation and discovery of best-of-breed selections in 8 content categories, for consumers seeking a hip customization service for their phones.

Smashing Ideas' & Blue Sky North’s advanced Flash wallpapers provide unique visual experiences which give Flash Lite enabled handsets a personal touch sure to impress and delight.



Smashing Ideas
Smashing Ideas has been the leader in creating rich media and Flash-based entertainment for more than eleven years. Smashing Ideas develops brand-building experiences and engaging content for many of the top entertainment destinations on the web, and has created more than 300 games for multiple platforms including PC, IPTV and Mobile.

Blue Sky North

Blue Sky North, a Smashing Ideas company, specializes in Flash design and development for mobile devices. Blue Sky North's award winning design and development service brings interactive applications and engaging games to the latest generation of Flash enable devices. The company's advanced Flash wallpapers including Street Skate provide viewers with unique experiences. Street Skate is an urban environment brought to life in the graffiti-filled skate animation with realistic motion and graphics.

MediFacts

In an emergency situation, vital facts can save time and save lives. Blue Sky North, a Smashing Ideas company, created MediFacts. The application lets users carry their most vital medical information in an easy to use application – right on their mobile phone.

Blue Sky North
Blue Sky North, a Smashing Ideas company, specializes in Flash design and development for mobile devices. In an emergency situation, Blue Sky North's Medi Facts application can help save time and save lives. Users carry their most vital medical information in this easy to use application – right on their mobile phone.

FishFeeder

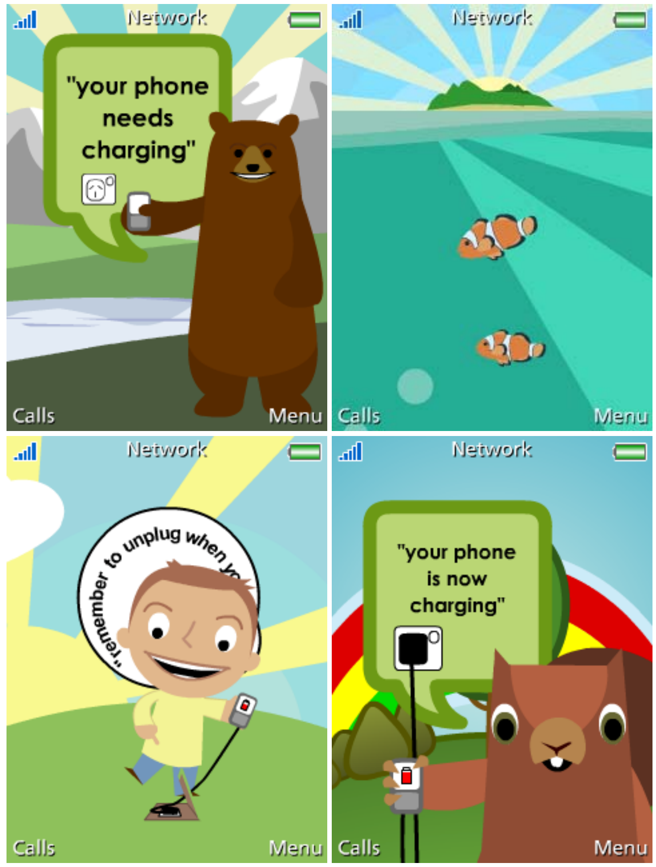
CELL's casual mobile game, Fish Feeder, delivers a highly immersive experience. The goal of the game is to successfully feed the fish so they grow into bigger fish. Animated fish swim across the screen as food drops from above. Players click on the button to get the food to land in the fish's mouth, successfully feeding three fish to move to the next level. There are three levels of play – each level represents a different fish species that is subsequently larger than the previous level. If a player passes all three levels a big monster fish swims into view.

CELL
CELL, a leading Japanese developer of mobile and commerce sites based on Adobe Flash Lite technology, is at the forefront of the casual gaming sector. The development teams at CELL are feeding Japan's mobile frenzy using Adobe software to provide applications and content to help network operators support increasingly sophisticated mobile services and generate more revenue. CELL's casual mobile game, Fish Feeder, delivers a highly immersive experience. The goal of the game is to successfully feed the fish so they grow into bigger fish. Animated fish swim across the screen as food drops from above. Players click on the button to get the food to land in the fish's mouth.

MoCarbon Animated Wallpapers

One of the biggest wasters of power around the world is mobile phone chargers. MoCarbon Wallpapers is a brand new, environmentally-conscious product from Moket. The animated wallpapers help customers to be more conscious of their recharging habits to help reduce wasted power.

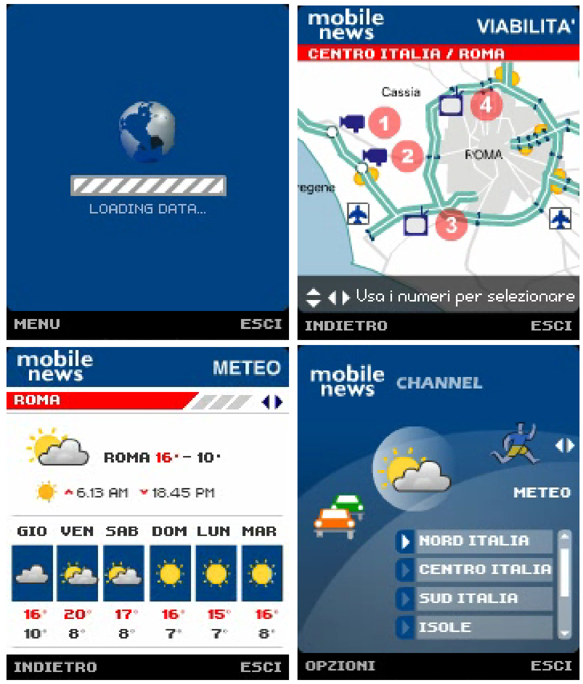
Moket
Moket develops ideas and content for mobile phones and devices based on the Adobe Flash Lite development platform. Mokets' Flash MoCarbon Wallpapers interact with a phone's battery and power source to notify and remind users of how to efficiently power their mobile phone.



Mobile News Channel

01design's Mobile News Channel is an interactive channel providing a variety of information to the end-user via text, images, and videos. World news, economic news, sports, weather and traffic are the main topics that can be explored through this information channel.

01design
01design creates multimedia B2B and B2C applications and content for mobile and Internet communication. Adopting a user-centered design approach and light technologies like Flash, 01design brings more than 10 years of Internet development to the mobile space. 01design's Mobile News Channel provides a variety of information to the end-user delivered through text, images, and videos.



Link: Sample Flash Lite Content
http://www.adobe.com/go/mobileexchange

Flash Lite versions

The following versions of the Flash LIte player exist in the market: Flash Lite 1.1, Flash Lite 2.0, and Flash Lite 2.1.

In 2007, many of the existing Flash-enabled devices are pre-installed with Flash Lite 1.1, while some support Flash Lite 2.0. With the release of the Flash Lite 3.0 player in Q4 2007—including support for FLV files—you will have more opportunities to create and distribute content. The penetration of Flash Lite 3 devices will grow over time as more device OEMs and operators adopt this new technology. In the table below, you can see how the different versions of the Flash Lite player compare to the desktop Flash Player 8 and 9.

Feature Support Comparison: Adobe Flash® Lite™, Flash® Player and Flash® SDK Comparison at a Glance

Audio	Flash Lite 1.1	Flash Lite 2.1	Flash SDK 7	Flash Player 8	Flash Player 9
MIDI	Device dependent	Device dependent			
PCM and ADPCM	✓	✓	✓	✓	✓
MP3	✓	✓	✓	✓	✓
8 Channel				✓	✓
32 Channel				✓	✓
Image & Video	**Flash Lite 1.1**	**Flash Lite 2.1**	**Flash SDK 7**	**Flash Player 8**	**Flash Player 9**
PNG	In authoring	In authoring & device dependent (1)	In authoring	✓	✓
JPEG	✓	✓	✓	✓ (prog JPEG)	✓ (prog JPEG)
GIF	In authoring	In authoring & device dependent	In authoring	✓	✓
Animated GIF	In authoring	In authoring	In authoring	In authoring	In authoring
MPEG-4 & other video formats		Device dependent	In authoring	In authoring	In authoring
Flash Video (FLV)			✓ On2 & Sorensen	✓ On2 & Sorensen	✓ On2 & Sorensen
8 bit alpha channel video				✓	✓
Import/Encode video			✓	✓	✓
Multimedia	**Flash Lite 1.1**	**Flash Lite 2.1**	**Flash SDK 7**	**Flash Player 8**	**Flash Player 9**
Dynamic loading of multimedia files		✓(2)	JPEG / MP3 only	JPEG / MP3 / GIF / PNG / Prog. JPEG	JPEG / MP3 / GIF / PNG / Prog. JPEG
Text	**Flash Lite 1.1**	**Flash Lite 2.1**	**Flash SDK 7**	**Flash Player 8**	**Flash Player 9**
Character set	Latin-1 & Shift-JIS	UTF-8	UTF-8	UTF-8	UTF-8
Complex languages (Thai, Arabic, Hebrew, etc.)		✓			
Dynamic text	✓	✓	✓	✓	✓
Device-specific vector fonts		✓	✓	✓	✓
Improved small text readability		✓	✓	✓	✓
Text measurement		✓	✓	✓	✓
Text wrap		✓	✓	✓	✓
Inline Text Input		✓		✓	
Predictive Text Support		✓			
Rich text styles			✓	✓	✓
FlashType				✓	✓
Improved text layout				✓ (justified, kerning, char spacing)	✓ (justified, kerning, char spacing)
Emoticons	✓	✓			
Emoticons in predefined color					
Interactivity	**Flash Lite 1.1**	**Flash Lite 2.1**	**Flash SDK 7**	**Flash Player 8**	**Flash Player 9**
Keyboard events	Device dependent	Device dependent	✓	✓	✓
Key-based navigation	✓	✓	✓	✓	✓
Mouse/Stylus events	Device dependent	Device dependent	✓	✓	✓
Mouse wheel support			✓	✓	✓
Programming	**Flash Lite 1.1**	**Flash Lite 2.1**	**Flash SDK 7**	**Flash Player 8**	**Flash Player 9**
Flash version supported	Flash 4 or earlier	Flash 7 or earlier	Flash 7 or earlier	Flash 8 or earlier	Flash 9 or earlier
ActionScript Version	FlashScript (Flash 4 or earlier)	AS 1.0, 2.0 (Flash 7 or earlier)	AS 1.0,2.0 (Flash 7 or earlier)	AS 1.0,2.0 (Flash 8 or earlier)	AS 1.0, 2.0, 3.0 (Flash 9 or earlier)
Dynamic loading of SWF data	✓	✓	✓	✓	✓
XML parsing		✓	✓	✓	✓
String/Array/XML-to-native-objects conversion		✓	✓	✓	✓
ActionScript strict mode		✓	✓	✓	✓
Set/Clear interval		✓	✓	✓	✓
Shape-drawing API		✓	✓	✓	✓
Ability to store data		✓	✓	✓	✓
Bitmap effects & filters				✓	✓

Bitmap caching				✓	✓
BitmapData API				✓	✓
Blend modes				✓	✓
9-Slice scaling				✓	✓
Stroke enhancements (endcaps & joins)	·			✓	✓
Linear & radial gradient enhancements	·			✓	✓
Ext. API for browser scripting	·			✓	✓
File Upload/Download	·			✓	✓
IME API enhancements	·			✓	✓
Other Features	**Flash Lite 1.1**	**Flash Lite 2.1**	**Flash SDK 7**	**Flash Player 8**	**Flash Player 9**
Generic browser interface	✓	✓	ActiveX or Netscape	ActiveX or Netscape	ActiveX or Netscape
Dynamic memory handling	✓	✓	✓	✓	✓
Device-specific capabilities	✓	✓			
Meta data support					
Background Transparency		✓ (set in host application)		✓	✓
Forward lock					
Printing			✓	✓	✓
Object model (for components)		✓	✓	✓	✓
Improved event model		✓	✓	✓	✓
XMLSockets		✓		✓	
FMS: RTMP Streaming			✓	✓	✓
FMS: Remote Shared Object			✓	✓	✓
Scriptable masks			✓	✓	✓
SWF file compression		✓	✓	✓	✓
Accessibility			✓	✓	✓
Dynamic discovery of device features		✓	✓	✓	✓
AS exception handling		✓	✓	✓	✓
Improved ActionScript perf				✓	✓
Improved rendering perf				✓	✓
Web services and SOAP API		✓	✓	✓	✓
New preloader API		✓	✓	✓	✓
Progressive download			✓	✓	✓
Basic version check and update				✓	✓
Express install				✓	✓
Enhanced local file security				✓	✓
SVG-T 1.1	✓	✓			
Flash Lite Features	**Flash Lite 1.1**	**Flash Lite 2.1**	**Flash SDK 7**	**Flash Player 8**	**Flash Player 9**
Access to device-specific features (volume, backlight, vibrate, and so on)		✓			
Launch native applications	✓	✓			
Reduced runtime memory consumption	✓	✓			
Graceful handling of out-of-memory condition.	✓	✓	✓		
Interruptible/Re-entrant player	✓	✓			
Runaway script limit	✓	✓	✓	✓	✓
ActionScript slicing	✓	✓		·	·
System Requirements	**Flash Lite 1.1**	**Flash Lite 2.1**	**Flash SDK 7**	**Flash Player 8**	**Flash Player 9**
Player size (core player DLL)	275K	450K	1.0MB (Win & Win CE)	1.4MB (Win)	~2.0 MB (Win)
CPU characteristics	32-bit data bus, 200Mhz ARM9	32-Bit data bus, 200Mhz ARM9	32-bit data bus, 300Mhz ARM9	Desktop Hardware	Desktop Hardware
Minimum RAM requirements	64K	128K	4.5MB	Desktop Hdwr	Desktop Hdwr
Recommended RAM		2 MB	32MB	32 MB	32 MB
Content size-to-heap ratio	1:10 (3)	1:15 (3)	1:30		
Platform/Browser	**Flash Lite 1.1**	**Flash Lite 2.1**	**Flash SDK 7**	**Flash Player 8**	**Flash Player 9**
Reference platforms	Symbian	Symbian, BREW	Win XP (SA, ActiveX), WinCE (Active X), Linux (SA, Netscape plug-in), PocketPC (ActiveX)		
Browsers supported			IE, Firefox,	IE, Netscape, Firefox, Mozilla, AOL, Opera	IE, Netscape, Firefox, Mozilla, AOL, Opera

Abbreviations: 'SA' = Standalone Player, Footnotes (1) Transparency supported (2) JPEG & MP3; other formats dependent on device-specific codecs (3) Estimated worst-case memory consumption: for example, for playback of 100k SWF file, the recommended memory configuration is 1.5 MB Source: Adobe - July 18 2007

Flash Lite, but what type of Flash Lite?

As soon as you scratch the surface of mobile computing you hit 'the matrix' – this matrix is an amazingly complex universe of mobile devices, operating systems, mobile carriers, OEMs, implementations, and versions.

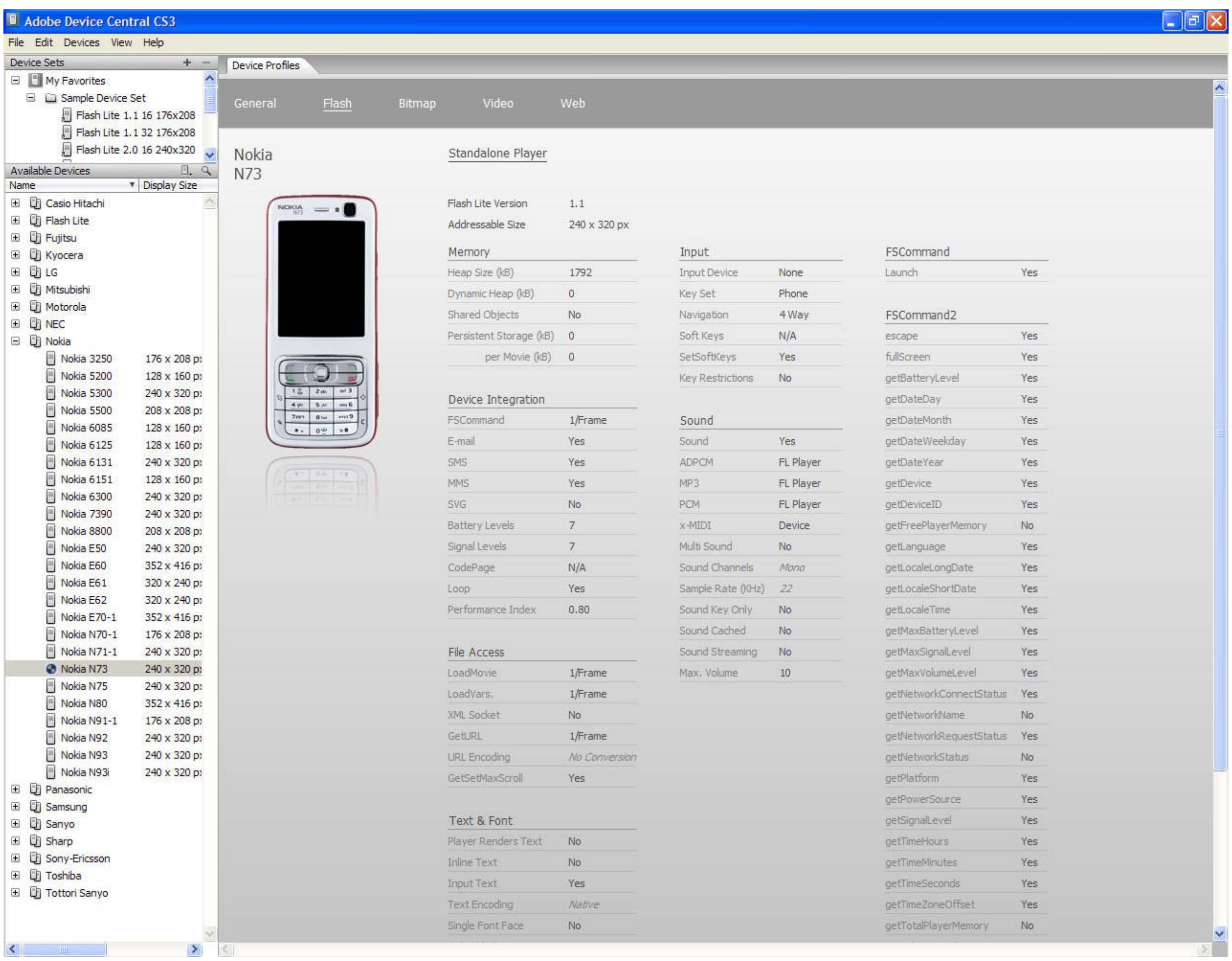
Flash Lite goes a very long way towards simplifying the way in which this universe of devices can be addressed by designers and developers, but there are some important distinctions regarding the different types of Flash Lite player versions and implementations that can be targeted by your Flash Lite content.

Another great help in understanding and addressing the matrix of mobile and non-PC devices in the world is Adobe Device Central CS3. Adobe Device Central CS3 is a testing environment for mobile content that is integrated with other Adobe CS3 products and which has a library of device profiles which is updated regularly. See more about Adobe Device Central at http://www.adobe.com/products/creativesuite/devicecentral/ and in this document here.

Each Flash Lite installation on a device supports one or more content types that the device manufacturer determines. For example, some devices use Flash Lite to enable screen savers or animated ring tones. Other devices use Flash Lite to render content that is embedded in mobile web pages. Not all content types support all Flash Lite features.

Each Flash Lite content type, paired with a specific device, defines a specific set of Flash features that are available to your application. For example, a Flash Lite application that is running as a screen saver is not typically allowed to make network connections or download data.

In Adobe Device Central, the Device Profiles tab shows what content types are supported for each individual device. Examples of content types are stand-alone player, wallpaper, and screen saver. For each content type that a device supports, the device profile shows relevant settings. When planning the content to deliver, consider the content types that a device supports.



For Flash Lite, the content type can determine the features that are supported on a device, and paired with the display size of a specific device, determines the addressable area on the screen. The addressable area, which can differ from the display size, is the maximum screen width and height in pixels for the content.

Note: For additional, up-to-date information about Flash Lite content type availability, see http://www.adobe.com/go/mobile_supported_devices .

Link: Current Flash Lite Content Type Availability by Carrier
http://livedocs.adobe.com/flash/9.0/main/wwhelp/wwhimpl/common/html/wwhelp.htm?context=LiveDocs_Parts&file=00005683.html

Why Do I Care?

Where Computing Occurs

Personal Computing is still in its infancy – the computer is only 66 years old. From punch cards and room sized computers to personal computers that fit under a desk, and now to today's mobile handsets with faster CPUs and more memory than personal computers had in the late 1990s.

The trend of mobile phone adoption coupled with Moore's Law means that personal computing is expanding / moving to mobile computing devices becoming more pervasive. Already the number of cell phones worldwide is far greater than the number of personal computers. Moreover, many people around the world - particularly in developing countries - will have their first Internet experience on a mobile phone.

By the end of 2007, there will be 2,713 million mobile devices in use worldwide compared to 755 million PCs.

Source for mobile devices is Gartner Dataquest (July 2007) and for PCs is Forrester (July 2007)

Flash Lite matters to today's designers and developers and their clients because Flash Lite offers the opportunity to extend companies' brand, message, and content to more users in a rich, consistent way and get it in front of end users in a more personal and more pervasive medium that commands users attention even more than TV or the PC.

Where Money Can Be Made

If you were to survey the history of revenue generated from content on the internet you would find that the majority of content that has been downloaded has not been paid for by the person downloading that content. Instead that content has been primarily paid for via ad revenue.

In contrast to that is the thriving ecosystem of content on mobile carriers' networks. Sales of content to subscribers is a growing percentage of carriers' revenue and is a multi-billion dollar business.

The benefit to the creator of content is that, unlike ad revenue that must be split across many creators in a sometimes esoteric and inexact process, there is a one-to-one relationship to each sale of a piece of content and revenue generated to its creator. In addition, mobile advertising is an emerging revenue source that, while relatively small today, promises to grow in the future.

See The Flash Lite Ecosystem – Show Me The Money in this white paper for more about the revenue opportunities with Flash Lite content.

The User Experience (UX)

Flash has an interesting history. It was originally a presentation technology focused on creating lightweight client-rendered animations. Thus the terms 'movie' for an SWF and the 'timeline' metaphor in the authoring environment.

To some developers or technical analysts these 'legacy' terms and the distant history of Flash as solely an animation technology bring them to the question, "Why use a 'timeline' to create a 'movie' when I want a software application?"

The answer to that question is "The User Experience". The rich rendering, animation, and multimedia capabilities of Flash Lite allow for any UX that can be imagined and which is possible on a given target device to be created.

In addition to raw functionality and features, creating an enjoyable and engaging experience for end users is the most important thing in software design and development. If the user enjoys interacting with an experience on a device then everyone from the manufacturer through the owner of the content to the wireless carrier is happier for it.

The ActionScript 2 programming language (supported in Flash Lite 2.x and later versions of Flash Lite) is a mature, strongly typed programming language that is based closely on the ECMA-262 programming language. The libraries or classes specified by this standard have been augmented to include many Flash-specific libraries which create a powerful event and object model.

C++, Java, and other low level technologies leave the developer to create their own rendering engine (almost always without good anti-aliasing), sprite engine, sound engine and they are generally dealing with raster images and text. The relationship between designer and developer is generally disconnected and sub-optimal.

With Flash Lite the reverse is true. The power of a mature presentation technology coupled with a solid programming language which has versatile event and object models allows the designer a much greater palette from which to work and does not limit them to what the developers can create for a presentation layer in 'short' order.

The Adobe Flash authoring environment is very intuitive to designers and Flash's object oriented Library facilitates changes by designers in parallel to work by developers.

Addressable Market: The Opportunity & Key Stats

As mentioned above, there is a thriving ecosystem of mobile content already in place. Flash Lite content has been a proven success with consumers in Japan where every mobile operator's network and nearly all handsets are Flash Lite enabled.

In order for a device to playback Flash Lite content, the Flash Lite player must be preinstalled on the device by the handset manufacturer or, in the case of Flash Lite 2.1 for BREW extension, be certified for download Over-The-Air (via the wireless network). The number of Flash Lite enabled devices has increased dramatically worldwide over the last couple of years.

As of June 20th, 2007 there were over 300 different models of Flash-enabled devices with a cumulative count of over 250 million Flash-enabled devices shipped worldwide.

An example of this solid growth in addressable market is Verizon Wireless' licensing of Flash Lite and related technologies. Verizon Wireless is the second largest carrier in North America and has been a leader in building a highly successful ecosystem around consumer content.

Flash Lite games, screensavers, and data enabled applications are now available on Verizon Wireless' network.

As mobile operators have started to enable Flash Lite on their networks and handsets, so have handset OEMs as well as other consumer electronic device manufacturers. Adobe projects that the number of Flash Lite devices will reach over 1 billion by 2010.

Link: Flash Lite Supported Devices:
http://www.adobe.com/mobile/supported_devices/

Technology Landscape: Competitors, Comparisons, and Coopetition

What criteria is used to identify a direct competitor to Flash Lite?

- Available across multiple handsets and devices
- Available on multiple operating systems
- Available on multiple mobile operators' networks
- Mature strongly-typed programming language
- Mature authoring environment for designers
- Mature authoring environment for developers
- Portability: Write once, tweak few, run everywhere
- Robust presentation layer
- Extremely compact file size – optimal for wireless networks
- Support for common video format across all handsets
    - Flash Lite 2.x & later: 3GP & MPG4
    - Flash Lite 3 & later: FLV (Flash Video)

When considering the full list of criteria above there really are no direct competitors that can match Flash Lite completely.

The criteria above that can't be matched by any potentially competitive technologies are:

- Robust presentation layer
- Support for common video format across all handsets (Flash Lite 3 only)

The criteria above which are only *barely* met by any potentially competitive technologies are:

- Mature authoring environment for designers
- Robust presentation layer

- Portability: Write once, tweak few, run everywhere

The closest thing to a direct competitor to Flash Lite is J2ME.

J2ME or "Java 2 Platform, Micro Edition" is the term that refers to the many implementations of the Java virtual machine designed for less powerful computing devices such as mobile handsets.

Java has not traditionally done very well on client computing devices, but instead has found wide adoption on the server.

One of the main reasons for this is the lack of a strong presentation layer built into the virtual machine. User Experience has traditionally not been a strong point of Java's.

Adobe creates reference Flash Lite players for OEMs to use to develop the Flash Lite player for their specific hardware. Before a Flash Lite player can be pre-installed and shipped on a device it must be certified by Adobe to meet the requirements of a Flash Lite player.

This careful facilitation, oversight, and certification of all versions of the Flash Lite player on all platforms ensures that developers can count on their Flash Lite applications running correctly on all Flash Lite players. This in part explains why adoption of a Flash Lite player version is slower on mobile devices than for the Flash Player on the desktop PC.

Cost Lite: Saving Time & Money

Making a superior UX for less time and money is a great strength of Flash Lite.

If you look at the evolution of game development for PC and gaming consoles you find that early games were developed in low level programming languages such as ANSI C and Assembly. When a bug was found in the game during testing the fix would require changes to the core code of the game and a complete recompile to test.
This was not time efficient.

Slowly the trend has been for game development companies to focus on building very strong 3D engines with robust object models and an optimized scripting engines. In this way aspects of the game play such as gravity, speed, and character strength can be fully described in a scripting language. When a tester finds a bug in a game built in this way the fix might be a 5 minute tweak of a value in a script file and it can be tested immediately without the need for the recompile of the entire game project.
Development in Flash Lite is very similar to this evolved game development method. The sprite engine, scripting engine, and the rest of the core Flash Lite technology is written in native code for the target platform. Feature implementation and bug fixes of a Flash Lite application, game, or other content is done via scripting and is a much quicker development process than fixing the underlying core capabilities of the application in order to achieve the fix.

The other huge bottleneck in software development is the interchange between developer and designer. In J2ME, for example, the designer generally works in a design tool such as Photoshop CS3 to create raster image assets for the Java developers to utilize in their work. Any animated transitions or interactive components of a J2ME application need to either be prototyped by the designer in a separate tool (often in the Flash authoring environment ironically) or described verbally to the developer to take a shot at implementing them based on the conversation. The developer meanwhile works in a separate tool all together.

When the transitions, layout, or interactive components don't quite turn out like the designer intended them to due to nuances lost in translation then it is time to perhaps tweak the assets and then certainly to force the developer into the position of trying to achieve the very important nuances by changing software code.

In the Flash authoring environment the designer and developer work in the same tool. Visual design, layout, transitions, and interactive components go through no translation at all. They are just as the designer created them. The developer works in large part with the actual assets that are kept in the FLA file's Library. When changes, bug fixes, or minor tweaks are needed in the User Experience the designer simply goes into the FLA file and makes the necessary changes.

The increase in product quality (UX) and decrease in development time with Flash Lite projects is easy to underestimate. A Flash Lite product's development time will take anywhere from 30-70% of the time the same product would take to develop in J2ME.

Concept to Cash: Design, Development, and Distribution

2

A solid pre-production process is key to the success of any software project. For teams that are newly moving into mobile content, it is all the more important.

There are considerations in mobile user experience that are particularly important.
Giving full thought to these key points of the mobile UX in pre-production will go a long way towards making the move into mobile software smoother and more successful.

Note:
This section has portions targeted at team environments and doesn't address sole contributor projects as directly. The concepts covered are still valuable for a sole contributor to consider.

Also, portions of this section are more applicable to deeper software projects and may not be directly applicable to more animation heavy pieces of contents such as wallpapers and screensavers.

Brainstorm, Design, & Prototype Iterations

Business Requirements: Why Before How

Often design and development teams get projects complete with a full list of requirements already specified. Sometimes not.

In either case it is important for the team to understand *why* this project is happening, *why* money and time is being spent on it.

When the reason for the product or project is known it empowers the team to make good choices in design and development.

Which features or aspects of the project are the most important ones?
If there is extra time in the schedule where should that extra time be spent?
Are there additional features needed to actually fulfill the overall objective of the project?

Acquiring this 'big picture' is a valuable first step to any project.

Brainstorm: What are the possibilities?

Some types of content, projects, or products don't really require a brainstorming session to kick them off. The requirements for these types of projects are so specific that there are not many options for the implementation.

Most projects, however, call for a more active and dynamic creative process.

The goal of a brainstorming session is to energize, resource, and kick off the creative process.

The quality and effectiveness of a brainstorming session are most directly affected by who is the moderator of the session and which people are chosen to participate.

The fruits of brainstorming sessions are:

- Possible starting points and inspiration for the UX and graphic designers
- Possible product features for clients and/or managers to consider

Good brainstorming sessions have these characteristics:

- A moderator with experience running a creative brainstorming session
- No debate, 'Devil's Advocate', or discussion of merits of an idea – all ideas are encouraged with no concern for feasibility or quality
- Building on other's ideas is encouraged
- Everyone is encouraged to participate

Design & Comprehensive Layouts (Comps)

From the brainstorming session comes ideas for different directions that the product could go. Decisions are made on which directions will be explored and presented via design comps.

The design team mocks up the different ideas and options. As with most creative processes the comps are likely to be an iterative process. Whiteboard sketches might be the first iteration followed by hand drawn sketches on paper.

There can be value in presenting general concepts to the customers and stakeholders in a low fidelity format such as scanned hand drawn sketches. This way no one is distracted by production quality but instead can focus on the design idea(s) being presented.

The comps are presented to the stakeholders for decisions regarding which aspects of which designs will continue into full production or 'signoff'.

The feedback from the customers/stakeholders might necessitate one or more iterations of the comps be created and presented to get final signoff on the design.

Important elements for final signoff:

- Branding elements
- Fonts
- Colors
- UI (User Interface) elements
- Transitions
- Navigational flow
- Key mapping

Model, Diagram, Document, & Develop

Model

While the design team is working on the design comps, there are many things that the development team can be doing in parallel: beginning to model the project, identifying existing classes from their library of existing code that might benefit the project, researching, and planning specific functionality.

Taking the time to model the classes that need to be included and/or created for the project prior to beginning implementation is an important step.

There are many modeling tools available to help with modeling classes including a free online tool at http://gmodeler.com which is specifically targeted to ECMA 262 scripting languages such as ActionScript and is built using Flash.

Link: Wikipedia: Class diagram
http://en.wikipedia.org/wiki/Class_diagram

Diagram

Managing expectations is important in collaborative projects of any sort. Making sure that the customers/stakeholders are expecting the product that your team will deliver is the most critical element for a successful project.

Diagramming is a very important tool in planning and making sure that all work done will be towards the desired end goal. Diagrams prove the old adage, 'A picture is worth a thousand words.'

Mobile projects especially benefit from good diagrams that show navigation, flow, and key mapping. Most mobile computing devices don't have touch screens or other pointer devices. Which keys on the handset's keypad will have which functionality is a very important aspect of a mobile application.

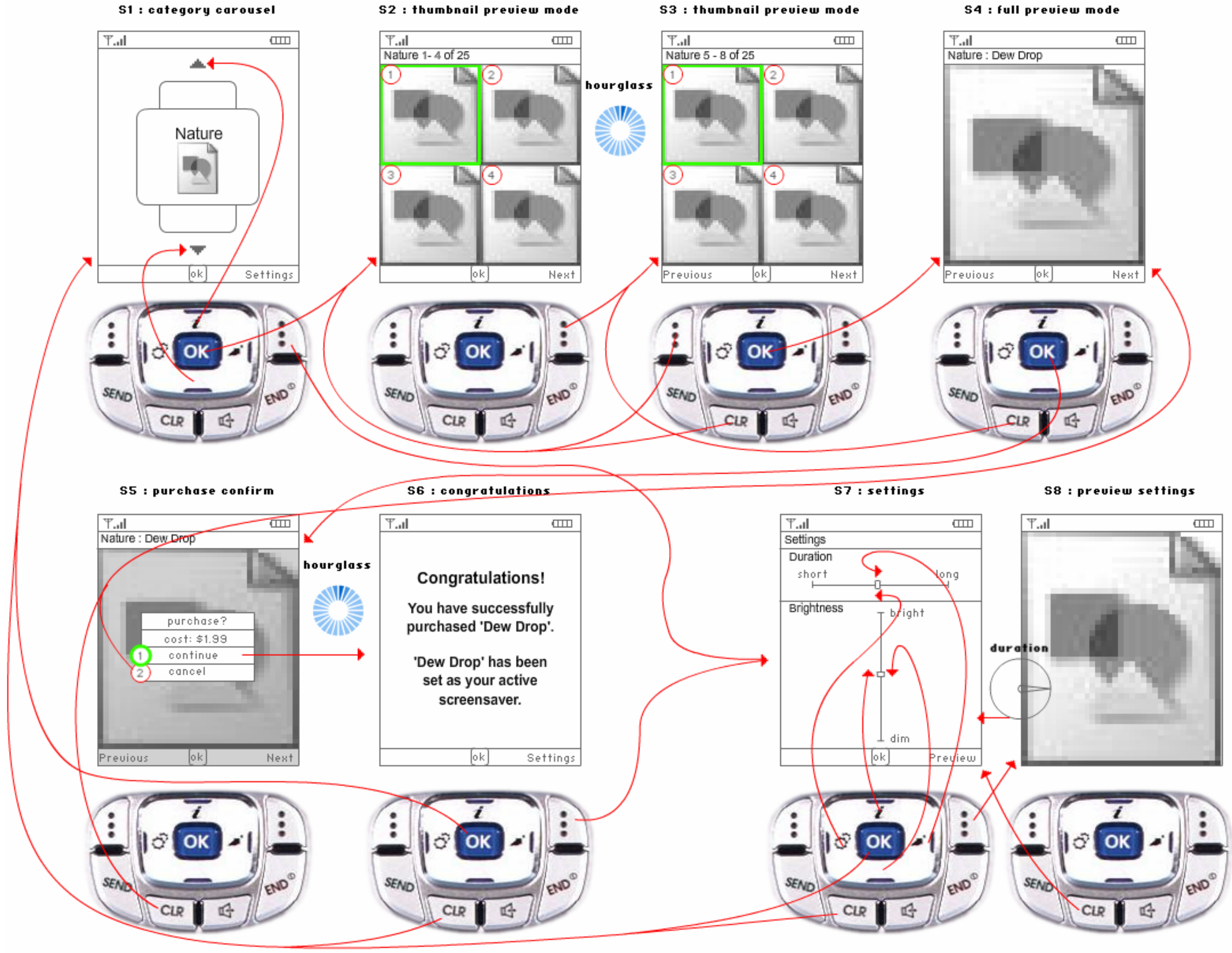
Communication to the customer/stakeholders is one function of good diagrams, but just as important is communication to the internal team. The diagram lets everyone on the team test their working model of the project against the diagram. Misunderstandings get cleared up, necessary questions get asked, and everyone gets to the proverbial 'same page'.

Even in sole contributor projects where the product is being created speculatively such that the designer, developer, and stakeholder are all the same person, modeling and diagramming are still very useful endeavors as they allow the creator to run into design conundrums in the planning stage rather than half way through development.

When a false expectation or a missed consideration is encountered after development has progressed much at all there tends to be attachment to not changing the existing code. This leads to less than ideal workarounds or design shortcuts and ultimately to a lower quality product with a sub-optimal user experience.

The example wireframe diagram below is for an early version of 'Smashing Screens' the screensaver catalog application developed by Smashing Ideas Inc.

Example: Wireframe w/ Key Mapping



Document

Documentation is the heart of a product development project. It is the touchstone for the customers, the project managers, the designers, the developers, and the QA team.

Functional Specification: What

The functional specification (functional 'spec') answers the question, "What will this product do?" in as close to laymen's terms as possible. It is sometimes important for a functional spec to also explain what the product *won't* do to avoid assumptions or false expectations.

Questions the functional specification must answer completely:

- What is the complete list of target devices for this product?
  - Considerations for different form factors (particularly screen size) should be listed here. Device Central is a great resource for different device specifications such as screen size.
- What is the reference device that will be developed to first?
  - It is a good idea to choose a reference device that is most similar to the superset of target devices with regards to CPU speed, memory, and screen size.
- Will this product be localized into other languages other than the original one?
- What are all the features that will be in this product?
- What are all the different step-by-step scenarios that must be handled by the applications – in other words what are the 'Use Cases' this product will be designed to cover?
- What are all the user interface elements that need to be created for this product?
- What are the distinct states that this application will be in?
  - It is good to have terms for these different states that the entire team can use to refer to them.
  - Screens: the states of the application that have a corresponding full screen user experience. A good term is 'Screen' as in 'Signup Screen' or 'Splash Screen'.

    - Modes: the states of the application that have subtler changes to the user experience. 'Modes' is a good term, as in 'Dialog Mode' or 'Buffering Mode'.
- What is the complete list of operative keys (and/or other user input methods) for each state of the application and what does each of them do in that state?
    - It is important to show when input that is otherwise handled uniformly in the application will not be handled in its usual way.
    - Differences across target devices should be listed here
- What visual design elements will be incorporated into this product?
    - The final comps that the customer/stakeholders have signed off on should be included here

Creating the functional specification is a collaborative effort of the entire team with the Project Manager (a.k.a. 'Producer', 'Program Manager', 'Project Lead', etc) acting as the scribe and facilitator who solicits the input of the key members of the team and owns creating the document, keeping it up to date, and publishing updates to all interested parties.

An example of this collaboration to create the functional spec is that the wireframe diagrams mentioned above may be created by the UX Designer on the project and be included in the functional spec by the Project Manager.

Technical Specification: How

The technical specification ('technical spec') answers the question, "How will this product be implemented?"

Questions the technical spec must answer completely:

- Which technologies will be utilized in this product?
- What are the different logical flows that will govern this product?
    - Flowcharts are really the only way to answer this question definitively
- What error cases will be handled by this product and how?
    - These error cases should be derived from the use cases in the functional spec.
    - Particular coverage of error cases due to loss of network (common in a mobile environment) should be addressed here.
- What classes need to be developed/included for this product?
    - This is where the class diagram from above is included
- If there is a server component of this application is there any synchronization between server data and persistent data on the client – if so how?
    - One-way or two-way synchronization?
    - On what schedule?
    - Full sync every time or only incremental changes?

Listing Assumptions

A good object lesson of where poor documentation and communication can get a development team is NASA's Mars Climate Orbiter mission, which missed its target embarrassingly due to one engineering team's use of the metric units of measurement and the other using the standard units of measurement. List assumptions no matter how obvious they might seem and make sure that all key contributors read the document.

Develop

Before & After Documentation

Some development can start before the specifications are complete, such things as basic common classes, for example key/input handling.

Another good common class to have is a layout class that can readjust the UI based on screen size (see Best Practices section below). The way to handle cross-device development is a business decision as much as it is a technical decision. Will there be differet installations/versions for each handset or just one?

Other than these common classes getting included into the project the main development should ideally start after the functional and technical specifications are complete. This is the application of the old adage, "Measure twice, cut once."

Parallel & Serial Development Tasks

Efficient use of the development team's time calls for a careful consideration of the distribution of work. What classes and features can be created in parallel? Which classes need to be created serially due to dependencies?

The class diagram showing interaction between classes is quite important here – allowing the development team to create classes in isolation to spec and have the different classes work together when incorporated into the project.

Most development projects have a Lead Developer on the project whose responsibility it is to make final decisions on architecture, assign development tasks to the different developers, and to be the liaison to buffer the developers from as many distractions as possible.

Targeted Devices & Device Central

One of the powers of Flash Lite is that it runs on multiple handsets and operating systems.
That can also be part of the challenge of developing a Flash Lite application.

Until Adobe's Device Central was available, the only way for a developer to test memory consumption, performance, and other important aspects of their mobile Flash projects was to load the application on the phone via USB or Bluetooth.

It is still necessary to have the target handsets to test on prior to shipping, but much time can be saved in development by using Device Central. Even though it is advisable to test any software on every target device it's not always feasible to test on all target devices given limited budgets and time.

See Device Central – A Mobile Developer's Best Friend below for more specifics on Device Central.

For more specific information on development for Flash Lite see Best Practices section below

QA & Test

The terms 'Quality Assurance' and 'Testing' are often used interchangeably, but they are actually distinct functions which are usually filled by the same team. It is best if the QA & Test Team are part of the project very early on, even during the early brainstorming sessions.

Testing is the general term for making sure that a product matches the specification – that the features are implemented as per spec and that the performance and stability of the product is acceptable.

Quality Assurance is the general term for making sure that the product provides an acceptable user experience. In QA the important thing isn't the specification but the subjective view of the end user experience.

Questions that get asked by QA:

- Is this experience cool?
- Does this product do what I would expect it to do? In the way I would expect it to?
- Is there anything annoying, cheesy, lame, or sub-optimal about this user experience? (notice the subjective nature of these terms)
- As I use this product in situations that an end user (in mobile software sometimes referred to as a 'subscriber') would, is there anything amiss?
  - These things can be subtle and subjective.
  - Navigational transitions – do they help the user have context within the application? Is the nuance of the transition pleasurable?
  - Wording, phrasing, and fonts – is all the signposting of the application clear and readily understandable to the general mobile user

Bugs found during QA are often challenges of the functional specification or design of the product. The QA team is charged with being an advocate for the end user and to bring up issues and concerns as they encounter them.

There are specific challenges to QA & test a mobile project. Compared to desktop computing where there are a few operating systems and a few browsers that most software projects have to take into account, the mobile computing world is much more complex.

Mobile Specific QA

Complexity Multiplied

The universe of mobile devices is an amazingly complex matrix of mobile device hardware, operating systems, mobile carriers, OEMs, implementations, and versions. It is nowhere near as simple as the software ecosystem on the desktop.

New mobile devices are released every month with new versions of new software packages and operating systems rolling out constantly. In comparison new browsers are released about every year and the same is true for the few operating systems that are widely adopted on personal computers.

With such a volatile software and hardware ecosystem there are few things that are standardized across mobile devices. Screen sizes are very nearly arbitrary (though some particular sizes are more widely adopted such as 240x320 – also known as 'QVGA'), keypads are often unique, and pixels-per-inch (PPI) is nowhere near standardization.

All of this presents a real challenge to QA & testing teams.

General Layout: Multiple Screens

Flash Lite scales the content proportionally by default to show all of the content in the area of the specific instantiation of the Flash Lite Player. This area is also known as "addressable screen size". The default scaling of the Flash Lite Player can be very useful for targeting multiple screen sizes.

There are other ways than the default scaling of Flash Lite to address the layout on different screens. These are described in Mobile UX Considerations below.

An aspect of QA and testing that is specific to mobile software is the multiple screen sizes that must be tested on. Device Central goes a long way towards helping preview the layout of an application on different screens. The one thing that Device Central really can't help with is to preview the different *pixels per inch* of a given device's screen.

Font Readability: Pixels Per Inch & 50 Year Old Eyes

The different physical size of screens and of their pixels makes font readability a very important consideration in QA. For example, many devices have a QVGA screen (or 240 pixels by 320 pixels); however, the physical screen sizes vary from 3.5 inches to 2.2 inches, which makes for some drastically different sized pixels.

Mobile users are not necessarily the same demographic as computer users. Designing mobile interfaces on small screens is all the more challenging because the eyesight of a the average fifty year old needs to be taken into consideration, as well as the fashion sense of the mall-hopping teenager.

The QA team may signoff on the layout and readability on one device, but not on another. See Mobile UX Considerations for tips on handling multiple screen sizes within one project.

Relative Performance

The Flash Lite player is excellent at accommodating variable CPU speed and memory resources. The nature of a cross-platform technology is that it does its best to be uniform across its supported platforms. The Flash Lite player does a great job of presenting the same SWF quite uniformly across multiple mobile devices and their different operating systems.

However, in some content such as certain types of intense action games, the slightest difference in performance can change what is an engaging and challenging experience on one handset into either a frustratingly difficult or simply boring experience on another handset.

The QA team needs to test the different content both in Device Central, which gives a good sense of the performance that will be experienced on the actual handset when calibrated to the specific PC it is running on, and on the targeted handsets themselves.

Loss of Network

Something that is quite unique to mobile and particularly wireless software is the need to account for sudden loss of network signal. Whether due to a tunnel, an elevator, or a dead spot in an operator's network, the reality is that the user is going to experience loss of network at some time.

The QA team can simulate loss of network in a few different ways (for example, tin foil, switching the radio on the handset off ('Airplane Mode'), or frequent visits to a handy spot where signal is lost (such as an elevator)) They should make a point of interrupting the network to gauge how well the application is handling this unfortunate eventuality. Adobe Device Central also has the ability to simulate loss of network (and level of signal strength) for the purposes of testing and QA.

Saving State: 10 Minute Sessions

Mobile computer usage patterns are quite different than desktop computing patterns.

While desktop computer users might sit down to have a computing session that lasts from 5 minutes to 5 hours, mobile computer users are far more likely to have frequent but short bursts of interaction with their mobile devices.

One overarching challenge for design and development teams getting into mobile content after much experience in creating desktop computing experiences is that the fundamental differences between the two user experience paradigms aren't necessarily apparent. The QA team can assure that the mobile user is not getting treated like a desktop user in any detrimental way.

An example of this would be a game where the user graduates from one level to the next in a series of levels of difficulty/achievement. If it takes 15 minutes to finish a level and thus progress to the next level and the mobile user is far more likely to play the game for 5-10 minute periods, then the game would become very annoying and soon be left alone as the user is feeling punished for not treating their mobile device as a desktop computer. In this type of case, QA might suggest that the progress of the user within any given level be saved at some interval so that the user comes back for their next short session on their mobile device and things are right where they left them.

The Flash Lite Ecosystem – Show Me The Money

The current mobile content ecosystem is established and complex at the same time. While the average developer can see their concepts turn to reality almost immediately in the online world, which has hundreds of publishing options, the same does not hold true for mobile. The Flash Player is ubiquitous in the online desktop world. It is synonymous with rich, engaging, digital experiences. Java and BREW® are ubiquitous in the mobile world, but Flash Lite is quickly becoming the technology that has the potential to shift multiple paradigms in this ecosystem: development, accessibility, and economics. With over 250 million Flash-enabled devices in the marketplace, and over 300 Flash-enabled device models available to consumers as of June 2007, the tipping point is near and Flash developers now have the ability to participate in one of the most lucrative industries in the digital world.

The reality of a mobile ecosystem that includes Flash Lite content/applications is quickly becoming apparent in several key territories worldwide. Flash Lite is currently distributed in the following channels (and often the same content is being distributed in multiple channels) in various countries around the globe:

On-Deck: This term is used for content that appears within an operator's mobile content storefront or portal (games, ringtones, applications, services), accessible only from the operator's handset. Operators such as NTT Docomo, KDDI, Vodafone, and Verizon Wireless currently distribute content in this manner.

Off-Deck: This term is used for content that may be purchased outside of the operator's mobile storefront/portal via WAP (Wireless Internet Protocol) mobile Internet sites, short-code promotion (a code that a mobile user sends via SMS in order to gain access to content via WAP push), or desktop-accessible websites, allowing for both m-commerce and e-commerce, via direct operator billing, PayPal or credit card.

On-Device Applications: Handset resident applications (sometimes referred to as "content portals") such as Nokia Content Discoverer and Sony Ericsson's PlayNow™ application that let networked users access programmed catalogs.

Embedded Content: Content pre-loaded on Flash-enabled devices, usually via a direct agreement with the handset manufacturer.

There are four key developments that are creating the surge for Flash Lite in the mobile ecosystem, which thereby allow the above channels to be viable. Each development listed below will explain the options available today to developers. They are the following:

1. The NTT DoCoMo phenomena
2. Embedded Flash Lite player on devices
3. The Flash Lite for BREW® extension
4. Emergence of Off-Deck and Alternative Channels

The NTT DoCoMo phenomena

Notably the most mature market for Flash Lite content, applications, user-interfaces and networked experiences is Japan, with NTT DoCoMo leading the charge in creating a turn-key B2C Flash Lite mobile experience. NTT DoCoMo's user base has reached the critical mass wherein a large percentage of its customers are consuming Flash Lite content on a regular basis, including the use of the new i-Channel FlashCast service which reached the 10 million subscriber mark in March of 2007. All new i-mode (NTT DoCoMo's proprietary mobile internet platform) handsets starting back with the 505i series mobile phones include Flash Lite, and nearly half of all i-mode mobile sites are Flash enabled.

As with any mature mobile content market, entering the Flash Lite content ecosystem in Japan is difficult. There are many established Flash Lite content/application developers, publishers and aggregators that have built the market over the past 4-5 years and have secured strong distribution partnerships. Distribution options include on-deck, off-deck via i-mode sites and other channels, on-device and embedded content. The most logical route to distribution in this territory is to work with one of the established companies in the market that can help launch your content/applications via off-deck/approved i-mode sites that are promoted by NTT Docomo, with the goal of building enough traction to gain visibility for on-deck consideration.

Embedded Flash Lite Player on Devices

More and more OEMs are embedding the Flash Lite player on their handsets. Throughout EMEA and APAC a large percentage of handset owners already have the Flash Lite player resident on their device, whether it's a Nokia or Sony Ericsson mobile device. This affords them the ability to consume Flash Lite content immediately. However, not all distribution channels currently support the SWF format, so the user must be diligent in finding appropriate sources. This will evolve though, so as with everything in the mobile world, the developer must be patient.

This embedded player allows operators to program their deck (or portal) with Flash Lite content for distribution as SIS files, the installer for Nokia S60 platform devices and as SWF files for Nokia Series 40 and Sony Ericsson devices, off-deck distributors to sell Flash Lite SIS and SWF content from their web and mobile/WAP sites, and OEMs to pre-install content that can be used immediately without a separate download required. As this handset base grows, so will the size of the opportunity for developers. Distribution channels will continue to emerge and realize Flash Lite as a viable content and application platform, increasing the demand for high-quality, original Flash Lite content/applications from the Flash developer community.

As the investment and time required on the part of OEMs to embed the Flash player is not trivial, it is expected that they will want to showcase this feature on their devices by including content that demonstrates the unique qualities of Flash Lite. This affords them the ability to differentiate not only their device, but also their content sensibilities for the target consumer demographic/s. Developers can either work with aggregators that have relationships with the OEMs or they can go direct to pitch their content/applications to an OEM. OEMs generally offer several options for compensation including, but not limited to the following:

- **No compensation but distribution:** Ability for a developer to see and promote their brand/content on millions of devices and have it accessible to all the device's users. This provides leverage for other channel distribution.
- **Fixed License Fee:** A fixed license fee that may require providing exclusive rights to the OEM for the content or application, depending on how high the fee. Some OEMs will offer non-exclusive deals, but those tend to be at a lower fee or a pure distribution/branding play.

For a list of the OEMs that currently embed the Flash Lite player, and their specific devices, visit: http://www.adobe.com/mobile/supported_devices/.

Flash Lite for BREW® extension

A critical component to building fast content distribution channels in a somewhat mature and established ecosystem is to find an established technology partner that can provide seamless integration into existing mobile operator infrastructure. That partner for Adobe is QUALCOMM and the advent of the Flash Lite extension for BREW® has afforded the Flash development community the ability to distribute their content/applications to an ever growing audience of customers on Verizon Wireless's network.

BREW® (Binary Runtime Environment for Wireless) is the technology developed by QUALCOMM that powers Verizon Wireless's (and other CDMA Operators') advanced data services. The Flash Lite for BREW® extension provides Verizon Wireless consumers the ability to download Flash Lite content/applications OTA ('Over the Air' which denotes transfer of files via the cellular wireless network), right from Verizon Wireless' Get It Now® service. Instead of having the Flash Lite player embedded on the handset, the extension allows for users' devices to be instantly "Flash enabled" upon the first download of a BREW® application with Flash Lite content. As of June 2007, there are 13 Flash Lite certified handsets on Verizon Wireless with the expectation that this will nearly double in the next 12-28 months. The represents a healthy percentage of the overall multi-media compatible handsets and therefore potential consumers in Verizon Wireless's network.

Developers have the following choices for getting their content to market on Verizon Wireless:

- Work with an existing Flash Lite aggregators such as the following:
  - Smashing Content
  - Shockwave
  - FunMobility
  - Mobitween
- Pitch their concepts directly to Verizon Wireless.

It is very important for a developer to realize what is required to become a BREW® developer, understanding the submission, testing and market requirements for their products. Of the above choices, the easiest path is to work with an existing aggregator as they will have established defined processes for bringing your content/applications to market and will have existing relationships with operators, OEMs, Off-Deck channels, or consumers, or all of the above (as there are only so many slots for direct deals with these distribution channels). Some of the key criteria to consider when reviewing your options:

Process to become a BREW® certified developer, a Verizon Wireless Developer and to register with NSTL

Information on the application process can be found at http://brew.qualcomm.com/brew/en/developer/overview.html, and http://www.vzwdevelopers.com/aims/. These steps will provide you with the necessary info and understanding of the BREW® environment, developer certification and the processes at Verizon Wireless. For reference, NSTL (National Software Testing Labs) is the established testing partner for Verizon Wireless, which handles True BREW® Testing of all content/applications that are considered for distribution via Verizon Wireless.

Porting—Is this really an issue?

Porting content/applications for mobile devices is a critical step in making sure that your product is ready for the mobile marketplace. Porting involves adapting your content or application for each target device's screen resolution, performance, handling of events such as sound, vibration, and flip event (to name a few), to assure a consistent experience across all target devices of your distribution channel. Although Flash Lite allows for a developer to create their content in such a way that the graphics will scale according to the device, it is important to consider all other elements that may affect the performance on specific handsets. Device Central will get the developer 70-80% of the way there, but that last 20-30% is critical to bring a product to market. What is revolutionary though, is the ability with Flash Lite, to create one to three SWFs that will cover all the FL certified handsets on Verizon Wireless, thereby saving thousands of dollars in porting costs. Keep in mind that that same file is generally portable with minor changes to all Nokia S60 and S40 Flash-enabled devices, plus several others.

Device Testing and final QA—the important last 20-30%

Device Central provides breakthrough tools as mentioned before, but the key to that last 20-30% will be to create MOD/MIF sets to test on representative devices including landscape handsets such as the LGVX9800 and VX9900. Not every device is the same and specific handsets such as the Motorola RAZR family of devices (including the V3C, V3M, and the K1M KRZR) and the Samsung A930, may provide programming challenges requiring the developer to scale back on their use of processor intensive features, unless they want to create builds that are specifically targeted for each device. Treat your QA as if not a single stone should be left unturned as any small functionality error that a developer may see, will always be caught by a testing house and internal testing at any operator. Hiring a 3rd party to do this testing is always a good idea before you submit to NSTL, as purchasing your own handsets and contracting individual plans for each handset (even corporate plans) can quickly become a very expensive venture. Working with an aggregator will help alleviate this cost and will provide an objective viewpoint on your content and how it functions on the device.

NSTL submission

Critical to the final distribution of your product (keep in mind that all content/applications should be pre-approved by Verizon Wireless before submission), Verizon Wireless requires all BREW® applications that access Flash Lite content to go through NSTL testing. Pure Flash Lite content/applications will still need to function in the BREW® environment and therefore some testing will be required either at NSTL or at Verizon Wireless. NSTL will test your content on each target device and such testing is very thorough. This is why it is of the utmost importance to make sure you have done exhaustive testing via Device Central and on the target devices before submitting to NSTL. Keep in mind that the NSTL submission process is very specific and it is extremely important that you are familiar with all the required information relative to your target handsets, their associated Flash Lite and brewSAplayer (BREW® stand-alone player) extensions the specific PIDs (Platform IDs) and derived platforms, and of course, the costs involved (specific information and detailed documents can be found on QUALCOMM's BREW® Developer Extranet site: http://brew.qualcomm.com/brew/).

Emergence of Off-Deck and Alternative Channels

Operators have dominated the mobile content distribution landscape for many years, looking to increase the ARPU (Average Revenue Per User) of their user base, at a time when voice revenues are declining at a rapid pace. Data revenues are paramount to the future profitability of operators, and represent the change in services and experiences that consumers expect from their devices. However, consumers are becoming more savvy and increasingly expect to be able to purchase and consume content from multiple sources (whether on the web, on-device, or via short-code).

Off-deck channels often supply multiple forms of content for consumers, and as these channels mature, they are beginning to offer Flash Lite content/applications. Examples of this emerging trend include distribution of Flash Lite content/applications via off-deck by the following:

- Handango
- Iguana Mobile
- Mobibase
- Mobitwee
- Moket
- Nokia Content Discoverer
- Smashing Content
- Sony Ericsson via Fun and Downloads and PlayNow™

A clear trend is to offer consumers content that they can both try in SWF format on a website, and then purchase via credit card, PayPal or direct carrier billing right from the website (or by being directed to a WAP site that will facilitate the download and transaction). Off-deck channels provide a new method of discovery, allowing for non-branded content a chance to compete in the mobile ecosystem, while offering established brands the opportunity to deliver a deeper, more immersive experience than that which is found on the operator's deck. The off-deck marketplace is most prevalent in EMEA (Europe/Middle East/Africa) and APAC (Asia/Pacific) with a fair percentage of all mobile content/application transactions taking place outside of the operators' decks.

Off-deck distribution of Flash Lite content/applications requires that the consumer's handset is already Flash Lite enabled (the same holds true for on-deck distribution in a non-BREW distribution channel). Unlike the OTA delivery of the Flash Lite extension for BREW® on Verizon Wireless, which instantly Flash enables the handset upon application download, Flash Lite content distributed via off-deck channels may only be targeted to those devices with the embedded Flash Lite player.

The distributed content/application must also be created in a version of Flash Lite that will match that of the embedded player, or in an earlier version such as 1.1, as all versions of Flash Lite are backward-compatible. For example, as of September 2007 there were 62 Nokia handsets that had Flash Lite pre-installed: a few models with Flash Lite 2.1, approximately 20 with Flash Lite 2.0, and the rest with Flash Lite 1.1 installed. As such, developers must gauge how important the feature enhancements of 2.0 and 2.1 are to their content/application, versus the breath of devices and thereby addressable audience that they can reach. In on-deck distribution, most operators will require publishers to port their content for the majority of their currently marketed devices, but off-deck channels may be more lenient (though revenue considerations will always play a part in this decision).

Creating A Distribution Plan

Now that you have an understanding of the opportunities in the marketplace, the next step will be to determine your distribution plan. There are very important issues to consider when building your plan, forecast, and investigating and signing distribution deals. In the mobile ecosystem, direct deals are few are far between. Consolidation is happening at a rapid pace as most operators are now relying on a select few aggregators and a handful of publishers to provide them with the content that they need. This holds true both in the United States and territories such as EMEA and APAC. When it comes to off-deck and distribution to OEMs, you may have better luck at developing a direct deal and this can be a good entry point for your products.

When determining where to invest your energy (and capital), it is advised to consider the following when selecting an aggregator or publisher:

- Does the aggregator or publisher have strong relationships with the distribution channels?
- Do they offer breadth of distribution in multiple territories?
- Do they require exclusive rights to your content/application, and does their plan for your content/application warrant the assigning of these rights?
- Do they offer a competitive revenue share that will allow you to realize a respectable amount of return?
- How many developers and properties do they represent? Will your content/application receive the appropriate amount of promotion and marketing?
- Does the aggregator or publisher understand Flash Lite from a development and market perspective, and will they be a partner in solving issues that arise during QA, certification, and distribution?

If you are able to sign a direct deal with a distribution channel, some of the key issues to consider are the following:

- What handsets are you required to address with your content/application?
- What testing and certification is required and what are the costs associated with these, including the purchase of target handsets and data plans?
- Are you required to have IP (intellectual Property) or E&O (Error and Omissions) insurance and what are the costs associated with this coverage?
- What type of technical and customer support must you provide the distribution channel? Will they require you to have 24x7 response to technical questions (many operators require this for their internal technical team)?

Your distribution plan should always include a roadmap for the amount of content/applications that you will need to supply in order to be an effective partner to the above listed channels. Most partners will want to make sure that you are in the market for the long-run and have a plan for your own success, plus have the technical aptitude and resources to bring your product to market. If you do sign distribution deals (either aggregator, publisher, or direct), remember that you will be required to deliver upon the terms of your agreement and most will have clauses that have remedies for breach of the contract.

Link: Mobile & Devices Developer Center, Sell your content
www.adobe.com/go/sellmobilecontent

The Tools: Flash CS3 Professional & Device Central

3

Note:

This section is not intended to be an exhaustive walk through of the Adobe Flash CS3, but is instead aimed at getting designers and developers who are somewhat familiar with the Flash authoring environment to comfortably create their first mobile application.

Flash CS3 – An Overview

Adobe Flash CS3 is the 9th version of the Flash authoring environment.

It is the first version of Flash since Adobe acquired Macromedia. This is significant because it is the first version of Flash that is integrated tightly and intuitively with the other tools that designers of digital experiences have been using all along – specifically Photoshop and Illustrator.

As previously mentioned in this paper, Adobe Flash started life as a presentation technology. The authoring environment still shows these roots by making any designer feel immediately at home. The tools that are most immediately apparent obviously enable the creation of visual elements.

Flash is also a comfortable environment for developers. The Actions panel where developers write their ActionScript code is newly improved in Adobe Flash CS3 Professional.

For a hands on step-by-step walkthrough of creating your first Flash Lite application see Hello Mobile World: Creating Your First Flash Lite Project below.

Links:

If you are new to Flash, you can learn how to Get Started with Flash at:

http://www.adobe.com/devnet/flash/getting_started.html

For more information on Flash CS3 features:

http://www.adobe.com/products/flash

Device Central – A Mobile Content Creator's Best Friend

Creating and testing mobile content has one key difficulty that desktop software doesn't – the large universe of devices that can be targeted and must be tested. Adobe has provided Adobe Device Central to greatly help creators of mobile content all the way through the life cycle of a mobile application.

Device Central is distributed and integrated with all the Adobe Creative Suite 3 editions and various point products including Adobe Photoshop CS3, Flash CS3 Professional, Dreamweaver CS3, Illustrator CS3, After Effects CS3, and Adobe Premiere Pro CS3. Designers and developers can now test their mobile content of these types: Flash Lite, raster images (bitmaps of different types), mobile web, as well as video content targeted at mobile devices.

Choosing your target device(s), previewing layout, monitoring memory usage, and previewing the performance that can be expected on the target device(s) are all features of Adobe Device Central. Since the mobile market moves so quickly, Adobe publishes device profile updates on a regular basis at Adobe Device Central Online (www.adobe.com/go/dconline).

To learn more about Adobe Device Central, go to www.adobe.com/go/devicecentral.

Getting Started: Tooling Up & Creating

4

Download This

- Adobe Flash CS3 Professional (includes Adobe Device Central)
- Adobe Photoshop CS3
- Check for Flash Lite Player and mobile updates to:
  - Flash CS3 Professional
  - Device Central CS3
- Download the latest device profile updates for Device Central on Device Central Online
- If interested in targeting BREW handsets:
  - Flash Lite for BREW Publisher for Flash CS3 Professional

Adobe Mobile Developer Program: Sign Up

Adobe's Mobile Developer Program has been put in place to support mobile device developers, designers, content and media professionals interested in mobilizing their Flash and non-Flash assets. With this program, you will have access to a broad range of services, information, and technical support to help design, develop, and market Flash Lite applications and content for mobile devices.

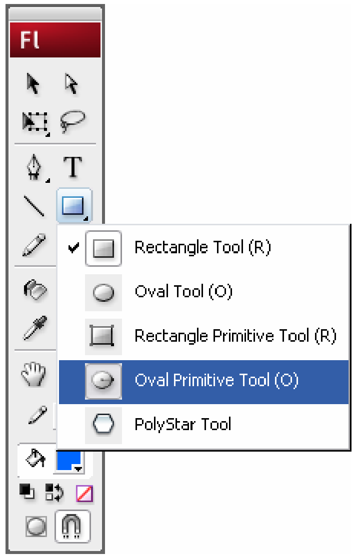
Link: Join the Adobe Mobile Developer Program:
http://www.adobe.com/devnet/devices/dev_program/

Hello Mobile World: Creating Your First Flash Lite Project

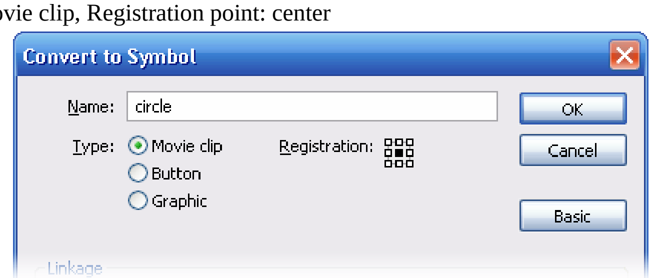
This section contains step-by-step instructions to walk you through the creation of your first Flash Lite Project.

This section assumes that you have downloaded and installed Flash CS3 Professional which includes Adobe Device Central.

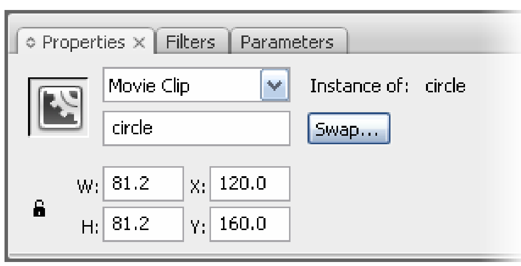
1. Start Flash CS3 Professional
2. File > New > Flash File (Mobile)
3. Adobe Device Central starts up to allow you to choose the main target device
4. In the 'Device Sets' panel choose 'Flash Lite 2.1 32 240x320'. This is the generic mobile device profile provided by Adobe for a Flash Lite 2.1 device.
5. Click the 'Create' button at the bottom right of Device Central
6. You are returned to Flash with a new document matching the dimensions and Publish Settings to match the target device
7. File > Save As… > save the FLA file as 'helloMobileWorld.fla'
8. In the Tools panel in Flash select the 'Oval Primitive' drawing tool by clicking and holding on the 'Rectangle Tool' and then selecting 'Oval Primitive Tool'.



9. Draw a circle near the middle of your workspace
10. While the circle is still selected choose the line and fill color in the Properties panel that suit your mood
11. Modify > Align > Horizontal Center, Modify > Align > Vertical Center
12. With your circle selected press the 'F8' key on your keyboard
13. Name: 'circle', Type: Movie clip, Registration point: center



14. In the Properties panel type 'circle' in the '<Instance Name>' edit box.



15. Insert > Timeline > Layer
16. Select the first empty Keyframe in 'Layer 2' on your timeline
17. Window > Actions (to show the Actions panel)

Copy the following ActionScript and paste it into the Actions panel

```
// Tell the player to stop on the current frame
stop();

/* Set the Stage's scaleMode to "noScale" – this will keep the Flash Lite
player from scaling any content
*/
Stage.scaleMode = "noScale";
// Set the Stage's alignment to Top Left
Stage.align = "TL";
// Set the rendering quality of the player to high
var qualStatus: Number = fscommand2("SetQuality", "high");
// Tell the player to display in full screen mode
var scrStatus: Number = fscommand2("fullscreen", true);
```

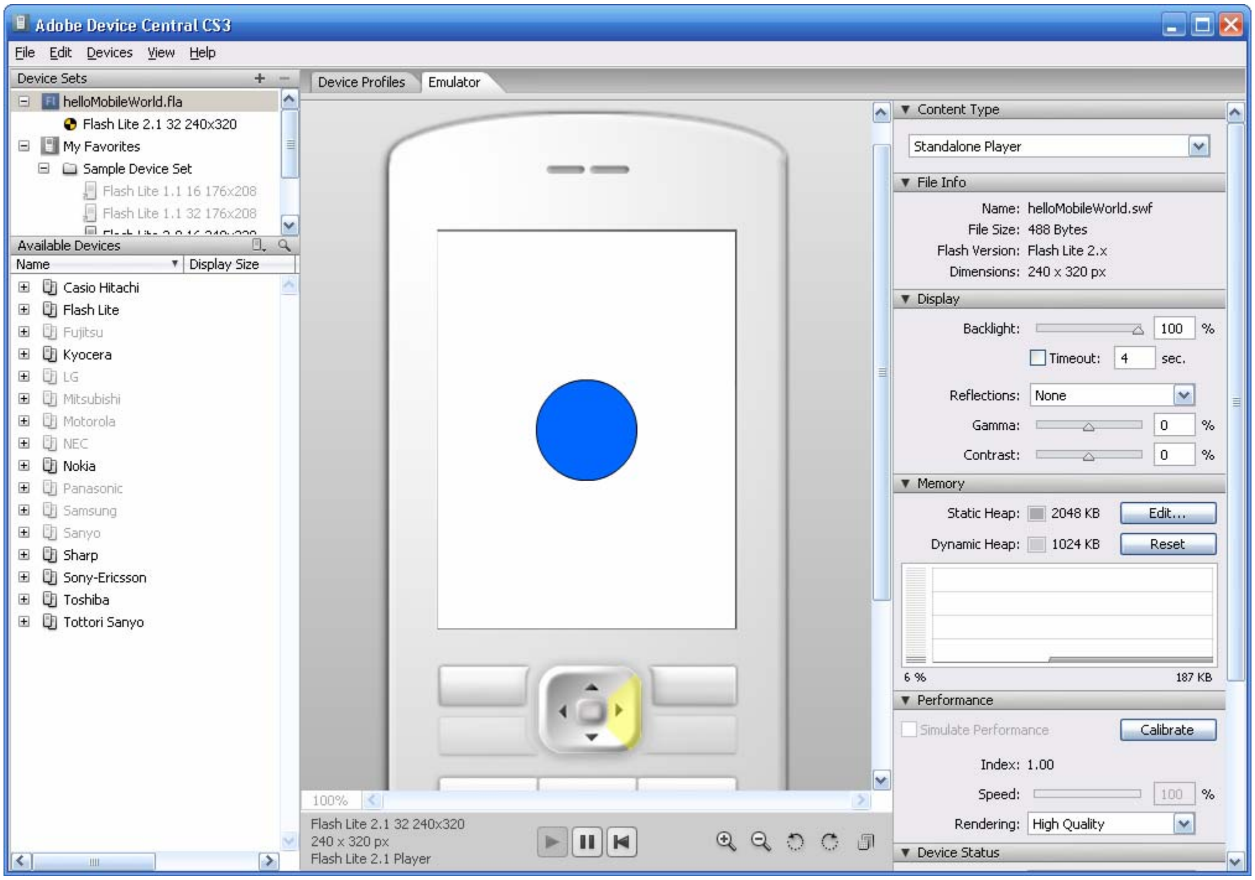
```
// Key Handling
Key.addListener(this);

function onKeyDown(): Void
{
     trace("onKeyDown");

     var keyCode = Key.getCode();

     switch(keyCode)
     {
            case Key.UP:
                   // move the circle up by 5 pixels
                   circle._y -= 5;
                   break;
            case Key.DOWN:
                   // move the circle down by 5 pixels
                   circle._y += 5;
                   break;
            case Key.LEFT:
                   // move the circle to the left by 5 pixels
                   circle._x -= 5;
                   break;
            case Key.RIGHT:
                   // move the circle to the right by 5 pixels
                   circle._x += 5;
                   break;
            default:
                   trace("\tunhandled keyCode:" + keyCode);
     }
}
```

18. File > Save
19. Control > Test Movie
20. Your first mobile application is shown in Device Central
21. Use the dPad direction keys (see 'Example: Keypad' below) to move your circle around the screen.


Adobe Device Central CS3
File Edit Devices View Help
Device Sets
helloMobileWorld.fla
Flash Lite 2.1 32 240x320
My Favorites
Sample Device Set
Flash Lite 1.1 16 176x208
Flash Lite 1.1 32 176x208
Available Devices
Name
Display Size
Casio Hitachi
Flash Lite
Fujitsu
Kyocera
LG
Mitsubishi
Motorola
NEC
Nokia
Panasonic
Samsung
Sanyo
Sharp
Sony-Ericsson
Toshiba
Tottori Sanyo
Device Profiles
Emulator
100%
Flash Lite 2.1 32 240x320
240 x 320 px
Flash Lite 2.1 Player
Content Type
Standalone Player
File Info
Name: helloMobileWorld.swf
File Size: 488 Bytes
Flash Version: Flash Lite 2.x
Dimensions: 240 x 320 px
Display
Backlight: 100 %
Timeout: 4 sec.
Reflections: None
Gamma: 0 %
Contrast: 0 %
Memory
Static Heap: 2048 KB
Edit...
Dynamic Heap: 1024 KB
Reset
6 %
187 KB
Performance
Simulate Performance
Calibrate
Index: 1.00
Speed: 100 %
Rendering: High Quality
Device Status

Mobile UX Considerations

Many of the UX (User Experience) considerations for mobile are the same as general UX considerations.

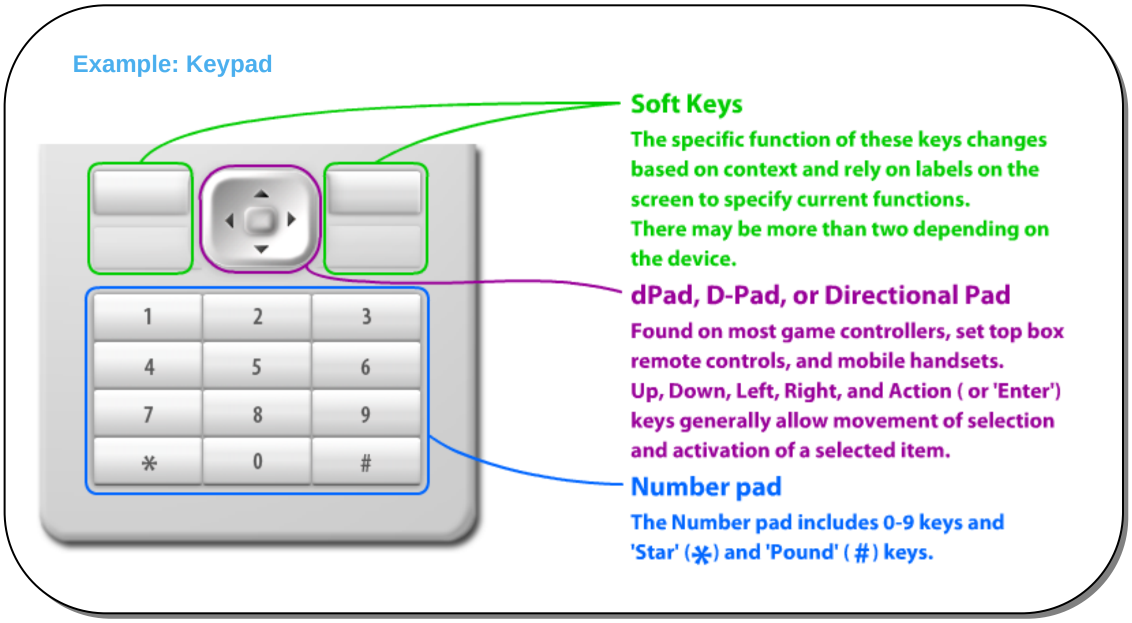
There are some specific points of consideration for mobile UX however that can make all the difference between a satisfying and effective user experience and a confusing and frustrating experience.

The first thing that a UX designer has to wrap their head around is the simple fact that a mobile computing device is NOT the same old personal computer but smaller, instead Mobile UX is quite different and there are new and very important considerations for designing an engaging, enjoyable, and efficient experience..

Below are the areas that need particular understanding and thoughtful attention by UX designers working on a mobile project.

Keys: A Pressing Issue

Most interfaces on mobile devices and set top boxes are driven by keys as opposed to pointing devices or touch screens. While there are devices that do have touch screens they generally also can be used via a keyboard. Interestingly the keypad on most phones maps almost directly to the keypad on most remote controls: numbers, direction keys, 'enter' key, often a 'back' or 'clear' key, and then some special keys such as soft keys on mobile phones – see the image below.



This means that solid code for the handling of key events is very central to any Flash Lite project. It also means that 'buttons', or really anything that can be selected and acted on, need to consider some different states than a desktop project might.

Selection States in Key Driven Interfaces		
State	**State Description**	**Button Analog**
Normal	The item when unselected	Up: Button without interaction
Selected	The item when selected – now it can potentially be activated	Over: Button when rolled over
Activated	The item when activated by the 'Action' or 'Enter' key	Down: Button while pressed
Inactive	The item when it cannot be selected	

Selection State: Where is the Focus?

Since Flash Lite applications are mainly driven by key presses as opposed to a pointers such as a mouse or stylus, it is of paramount importance that the user always know which UI element is currently selected and will be acted upon.

The indication of which item is currently selected is also known as ***the focus.*** The main design consideration around focus and indication of selection is consistency. Consistency in behavior, color, and type of indicator is what enables the user to always know which UI element has the focus. The measure of a projects usability around selection state is to take an uninitiated user, someone who does not use mobile phones much at all, and have that person use the application a bit. Ask them while they are using the application, "Ok, so what is selected now?" or "If you press the middle key on the dPad here, what do you think will happen?"

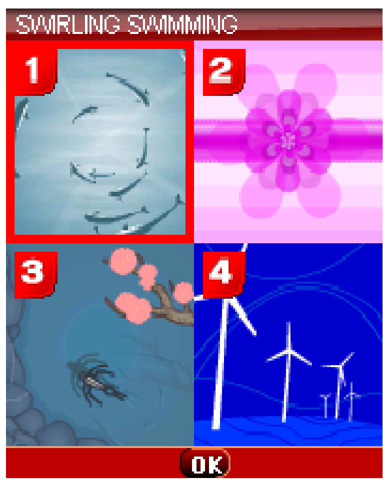
Asking uninitiated users their impressions is an example of an informal focus group. Focus groups are a form of qualitative research where a group of people are asked about their attitudes towards a product, service, or concept. Focus groups are often used as a form of usability testing which measures how well people can use an application for its intended purpose.

Even if a project has no budget for formal usability testing research it is still valuable to do your own informal research with friends, family, and whoever else might be at hand.

The image below shows the application 'Smashing Screens' and is a clear example of a focus indicator.

Example: Focus and Hotkeys

The 'Swirling Swimming' thumbnail is *selected* thus pressing the center dPad key will it. This application also has hotkeys –keys which will *activate* an item directly without having to first select it. The number icons show which key on the numeric keypad will activate which item.



Font Size & Pixels Per Inch

In web design it is easy to forget that not every user has the same nice monitor with the same high resolution as the designer.

In mobile design it is a fundamental starting point to know what screen sizes are being developed for, however an easy thing to forget to consider is the PPI or "Pixels Per Inch". The PPI tells you literally how large one pixel is in the human experience.

Example: Comparative Pixels Per Inch

The two handsets shown below both have 240 pixel x 320 pixel screens. The Sony Ericsson SO902i has a 1.9" screen while the Nokia N73 has an approximately 2.4" screen. The Nokia N73 has fewer pixels-per-inch, or in other words, it has larger pixels than the Sony Ericsson SO9021.

The design element most affected by PPI is text. The readability of fonts can be heavily impacted by smaller PPI ratios and so the PPI of the target device(s) must be taken into consideration by the designers.

Mobile device users are a varied demographic – even more so than desktop PC users. This means that older people with poorer eyesight need to be considered in the design of mobile applications. Fonts that are too small on the handset but which looked clean and slick on the designers' monitor during design do not create a good user experience.

Contrast & Readability: Flash Lite in the Light

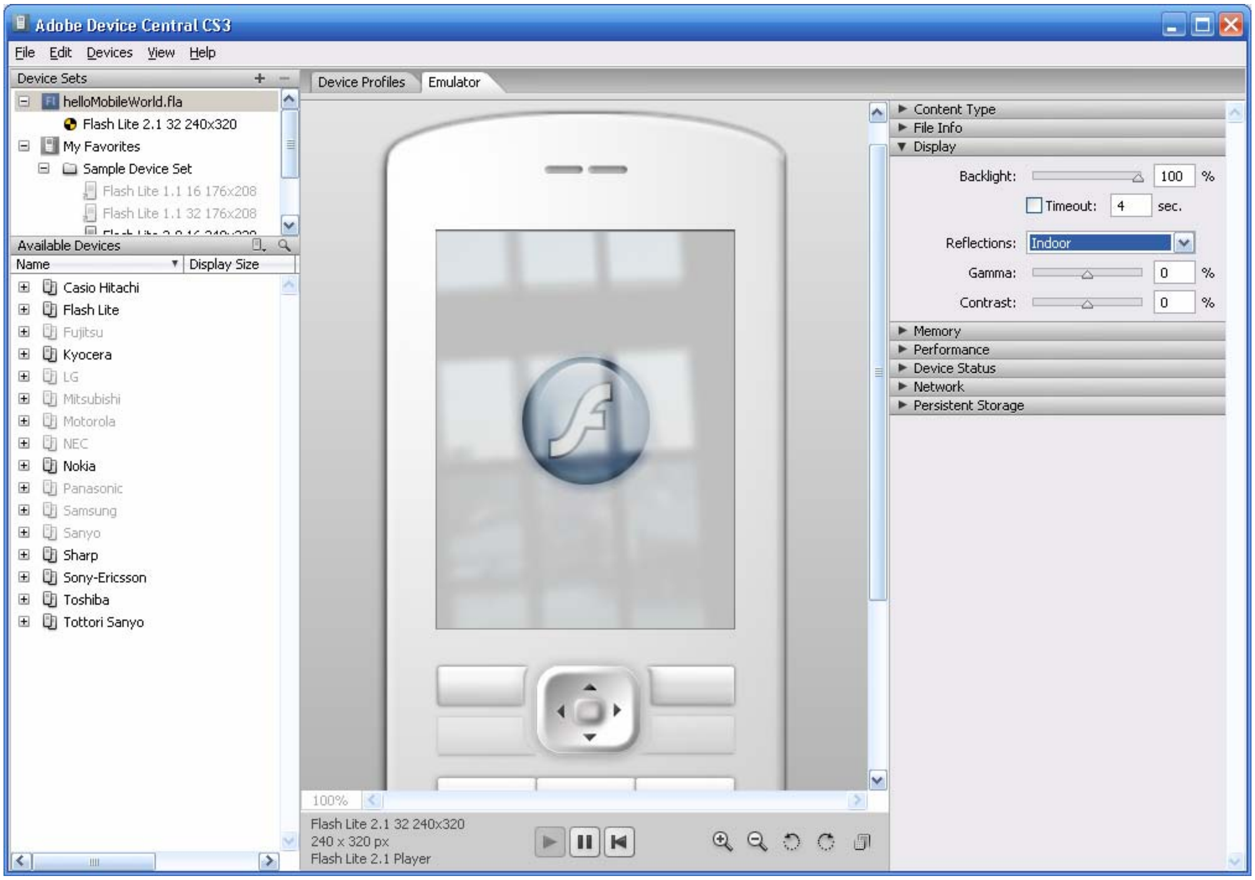
In desktop computing the monitor of the computer is usually positioned such that there isn't direct light on it and little or no glare or reflection.

Mobile computing is just that – *mobile*. The screen of a mobile device can and will be just about anywhere with all imaginable lighting conditions.

High contrast colors and good sized fonts go a long way towards minimizing the effects of glare and high light situations. Each handset has native fonts that can be used and, of course Flash can embed any fonts needed as well. Designers should experiment with different fonts to find the best option(s) for each project.

This is an area that Adobe Device Central can help a mobile designer with quite a bit. In the 'Display' panel a designer can adjust the backlight, gamma, and contrast of the screen as well as have reflections superimposed on the screen to test how usable their design will likely be in these types of lighting situations. As with all emulators, no matter how well made, this will only give an approximation of what to expect on the actual device and does not replace the testing of the application or content on the actual devices.

Below the 'Indoor' reflection has been selected to have the faux glare superimposed on the screen.



Long List Navigation: How Many Presses Does It Take?

Unlike desktop computers that have more room and multiple ways of navigating long lists mobile devices generally only have keys.

If a user encounters a list of 400 items on their handset and wants to get ¾ of the way down the list will they be forced to hit the down dPad key 300 times?

There are ways to mitigate the long list in mobile software and here are a few:

Key Repeat

Some mobile devices have built in key repeat and some do not. Of the devices that do there is a variance on the interval between key press events and how long a key must be pressed before it starts repeating.

Being able to press and hold a key down to get a certain action to repeat can be very useful to the user. Pressing and holding the down arrow to have a list scroll quickly is much preferable to having to press the down key repeatedly – once for each line of scroll.

Stationary Middle Selection

Once a list is scrolling the question for the user becomes, "When should I stop scrolling?" Since up and down arrows in a scrolling list need to both move the selection (in other words select the next or previous item) *and* scroll the list the UX designer needs to decide where the selection will be placed when the scrolling starts.

There are some strong arguments for leaving the selection in the middle of the visible portion of the list. This gives two benefits to usability:

1. Anticipation: It allows the user to see what next and previous items are before selecting either. In contrast, when the scrolling of the list only begins after the last visible item is selected then the user has no idea what is going to be selected next until it is selected. That is to say that the first time that the item is visible it is also selected, so there is no opportunity to anticipate. See the image below for an example.
2. Bookends: If when the list is first presented the first item is selected at the top of the list and a few clicks moves the selection down a few lines before scrolling commences as the selection stays stationary then the user knows to anticipate that the selection will start to move down the visible items when the end of the total list is nearing.

Carousel Lists

A list which 'carousels' allows the user to jump the focus (selection) from one end of a list to the opposite end (for example the focus would jump from the very last item in a list to the very first item) by continuing to press the directional key that navigated to the last item.

Consider this example use case:

1. The user is moving the selection in a vertical list of items using the down dPad key (the image below depicts this)
2. The user has scrolled to the bottom of a list of items such that the last item is now selected
3. The user hits the down dPad key again – what should happen?

Some user interaction designs would have a 'hard stop' where nothing happens in the case above. A design that allows for carouseling would enable the selection to jump to the top of the list in the case above.

Carousel lists can be very useful in decreasing the number of clicks required for certain use cases.

Hyper-Scrolling & Region Indicator

There are a couple of UX elements that can be quite useful in the instances where there is a very long list that must be navigated on a mobile device, key repeating has been enabled, and the list is alphabetical.

Hyper Scrolling: When key repeat is enabled on a device there is an initial delay after the key is first held down until the key starts repeating. Once the key is repeating and that key is controlling the scrolling and selection in a list then there can be a secondary level of increased scrolling – Hyper Scrolling. This only really makes sense when you have a very long list such as 100 items or more. Instead of each key scrolling the list by one item it can scroll and increment the selection in the list by more than one. This allows rapid, but gross navigation within a list and can be quite useful in very large lists.

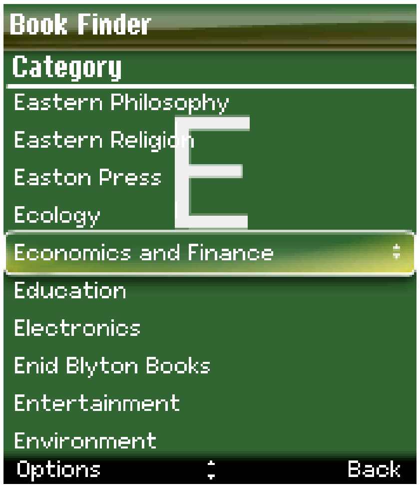
Region Indicator: When a list starts scrolling quickly either due to key repeat or Hyper Scrolling it can be a very useful thing to use a Region Indicator. This is when a visual cue is displayed to show the user what section of the overall list their current selection is within. In an alphabetical list it could be as simple as a large letter to let the user know that they are currently traversing the portion of the list that starts with 'E' for example.

Hotkey Linking: Another useful method for navigating long alphabetical lists is to enable the keypad to jump to the corresponding portion of the list. Where the '2' key on the handset has 'a', 'b', and 'c' alpha characters mapped to it then successive presses of the '2' key would jump the user to the portion of the list that starts with each letter. The region indicator would be presented momentarily as a signpost of the navigation that has just occurred.

Example: Hyper Scrolling, Middle Selection, and Region Indicator

The image below is a mockup of a mobile application called 'Book Finder'. In this example the user is scrolling through all of the book categories.

Key Repeat is enabled, it is a long list, Hyper Scrolling has been enabled, and you can see that the current selection is stationary in the middle of the visible items as well as a Region Indicator in the form of the superimposed 'E'. Also, this list carousels – hitting the down dPad key when the bottom item is selected will jump the user to the top of the list as opposed to making them scroll all the way back up from the bottom.



Transitions: Where Am I Now & What Just Happened?

Mobile applications often ask the user to traverse a hierarchical model – navigating from one screen to another while diving deeper into some sections of the application or moving between modes of the application.

When going 'deeper' into an area of an application or moving from one mode to another within the application it is important for the user to always feel completely oriented and cognizant of where they have come from and where they currently are within the application.

Animation can be a very powerful tool to achieve this awareness of state/mode/location within an application. In western cultures writing is left-to-right and top-to-bottom. Thus for software targeted at those cultures transitions that move the user from one mode to another can employ left-and-right animated transitions to great effect as well as up-and-down animated transitions to let the user know that they are diving deeper into a particular mode of an application and then coming back 'up' from that deeper experience. For applications which are targeting cultures where writing is right-to-left (sometimes known as 'bi-directional' language or 'bi-di' for short), such as Hebrew, Arabic, and Thaana/Thâna, animated transitions should move in that direction. Flash Lite 2.x and greater support bi-directional languages in text fields.

The TiVo and iPod are both good examples of using animated transitions to let users have awareness of 'where' they are in their experience.

Remember The Memory: Mobile Code Optimizations

Mobile devices have limited CPU speeds and memory capacities. Whereas the average desktop computer CPU speed is around 1 GHz and the available memory is around 512 MB this is not the case for mobile devices. It is difficult to pinpoint any average across all mobile devices or smartphones (phones with significant CPU, memory, and with operating systems that allow 3rd party applications to run) but, here is an approximation that is very close (these numbers will continue to increase over time): 300 MHz CPU with 50 MB of memory.

Classes of Mobile Handsets

Among mobile handsets there are different device tiers which break down along lines of hardware and capabilities.

At the lowest end are the RTOS (**R**eal-**t**ime **O**perating **S**ystem) phones or so called "voice only" phones which only have the software stamped into the chips at time of manufacture and they are mainly only capable of receiving and sending calls with perhaps some token contact application or similar.

This consumer-focused smartphone, with a voice-centric form factor (one-handed use should be possible) is marketed to users primarily as a consumer multimedia device (for example, music, pictures, gaming, browsing and e-mail— Post Office Protocol [POP] 3).

Enhanced phone: A voice-centric mobile terminal with enhanced features, such as camera, MP3 player, video player, Java support, and calendar and contact synchronization. These devices support data services, such as Web browsing and multimedia messaging. Examples are the SonyEricsson S700; Motorola V600; Nokia 6230. See also Basic Phone, Basic Smartphone, Enhanced Smartphone, and Cellular PDA.

At the other end of the spectrum are the high end smartphones. The cost of the handset goes up with each tier, as such the handsets which aren't 'High End' are considered 'Mass Market' handsets as their price point makes them more ubiquitous in the marketplace. As shown in the table below (organized from low to high end tier) Flash Lite covers the high end smartphones and the upper portion of the 'Mass Market' handsets.

Mobile Handset Classifications		
Class	**Description**	**Flash Lite Support**
Voice Only - RTOS	RTOS (Real Time Operating System) powered handsets with closed operating systems – no viable path to install new software.	No
Low End Feature phone	Voice-centric handset with basic CPU and limited memory. Some limited capability to install software, usually with a color screen.	Not Yet
Mid-Range Feature phone	An open or proprietary operating system with CPU and memory sufficient to run highly optimized software such as Flash Lite. Includes consumer-geared multimedia features such as camera, music, gaming, and e-mail. Limited memory space for installed 3rd party applications.	Yes
High End Smartphone	A near PC-like processing ability with a faster CPU and more memory than lower tier devices. Open operating system designed to support many 3rd party applications after the handset has shipped.	Yes

One of the key differences between mobile devices and desktop computers is that most mobile devices are firstly communication devices which must honor this key function before anything else – particularly if you consider emergency situations. It wouldn't do to have a software application lock up a mobile handset so that calls cannot be made or received.

Memory

The memory situation is even more limited than what it might appear from the numbers above. Every Flash Lite implementation has a fixed-size static memory heap whose size is defined by the OEM. Flash Lite implementations may have an additional dynamic memory heap to utilize but this is up to the OEMs. Memory gets allocated to the static memory heap first and the dynamic memory heap (if present) second.

The best resource for understanding Flash Lite's memory handling and best practices therein is the paper linked to below by Prashant Panigrahi - *Memory management and optimizations in Flash Lite.*

Link: Memory management and optimizations in Flash Lite by Prashant Panigrahi
http://www.adobe.com/devnet/devices/articles/memory_management.html

Performance vs. Footprint: Vector vs. Raster

Aside from good memory management practices there are other ways to optimize your Flash Lite applications.

There is a bit of a "rock and a hard place" aspect of mobile development. Often the practices that limit your file size and memory usage will tax your CPU at runtime. Conversely the practices that make for the best performance at runtime can be the worst practices for footprint and memory utilization.

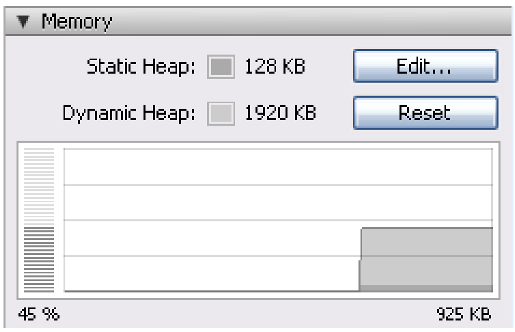
A case in point is the usage of raster images (bitmaps) vs. vector images. Raster images are pre-rendered images that take very little processing time to render at runtime. Vector images are exactly the opposite – mathematical representation of lines and fills that need to be interpreted and fully rendered at run time.

Raster images take up relatively large amounts of memory on disk and more heap memory at runtime when the image data is decoded for display. Vector images take up very little memory either on disk or at runtime. Vector images must be quite complex before they approach the size in memory of a raster image.

Memory is a precious commodity in mobile development – once you have used up your total allotment it is "game over" – your application locks up and the user is presented with an error dialog letting them know just that. This is one of the golden gifts of Adobe Device Central – the ability to track the memory usage of an application using the Memory panel which shows the usage of Static and Dynamic heaps while the application is running. As with all emulation the memory usage and thresholds are approximations and should not replace testing the application on the actual target device(s).

Example: Memory panel in Adobe Device Central CS3

The Memory panel allows you to see the application's *current* memory usage by static and dynamic heap. This is a crucial feature for developers and designers to optimize their code and assets for mobile devices.



While memory is the metaphorical 'rock' – the processor is the 'hard place'. Not only can the user experience slowness and choppy performance if you ask too much of the processor, but if you ask too much of the processor for *too long* your application will lock up and the user will be presented with an error dialog stating that the ActionScript is stuck.

An example of this is the processing of large XML files (Note that the processing of XML is only supported in Flash Lite 2.x and greater). Usually this parsing is done via a recursive function that walks the entire XML document and all of its nodes. Due to the low level optimizations for XML parsing in the Flash Lite player it is possible to line up enough calls to the parser that the device's CPU doesn't get a chance to send some cycles around to the other important processes running on the mobile device – such as the phone software.

The way to avoid tying up the CPU for too long is to use intervals to portion out the parsing into 'bite sized' pieces. Your interval parses some number of nodes, then continues on the next interval until the end of the XML document is reached and your interval can be cleared.

A Word on XML

XML has become the de facto method to share structured data across the internet. There are many wonderful things about XML, but arguably the best thing about XML is the wide adoption it has.

However, XML is not necessarily the best data transmission format for mobile devices. It is relatively verbose and therefore larger than it needs to be which makes transmission and storage of XML relatively costly. Also, once transmitted the XML must be processed and that takes CPU and memory resources much greater than a more efficient data format.

Many XML based web services have APIs that are not optimized for mobile computing. There is an increasing demand to make these web services consumable by mobile devices. There are instances where the sheer size of the XML document that comes in response to a call to such an API is sufficient to run the Flash Lite player out of memory.

There are three main methods for successfully consuming web services which have not been optimized for mobile and which use XML as the data transmission format.

1. **Proxy Server**: Implement a server-side component of the application that consumes the XML of the web service and processes it into easier-to-digest structured strings, name value pairs (useful for Flash Lite 1.1, or even just more succinct XML.

2. **Change the Server**: Work with the provider of the web service to implement some mobile-friendly APIs or to increase the flexibility of their existing APIs to enable more brevity in the responses.
3. **XML as Structured String**: Interface directly with the web service from the Flash Lite application but take the raw data (String) of the XML responses and process it as a String instead of as an XML Object. Serious memory and processor savings can be achieved by this method.

It is important to note here that there are many XML based web services that are completely mobile-ready and which can be consumed directly by Flash Lite without any of the above workarounds.

Some Flash Lite 1.1 Best Practices

There are millions of Flash developers worldwide and more developers are learning Flash development all the time. Mobile developers who are new to Flash and who get their first impression of Flash from working on a Flash Lite 1.1 project would be well served to take some time with Flash Lite 2.x as well to get a more complete picture of Flash Lite as a technology.

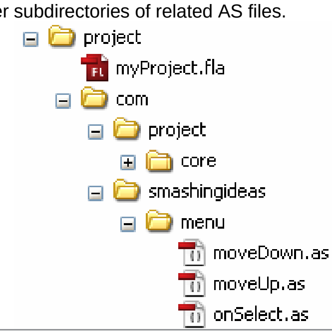
Extending the MovieClip Class in FL 1.1

Flash Lite 1.1 doesn't have classes or functions per se, however it is quite possible to organize your ActionScript and your FLAs in such a way that you are effectively extending the MovieClip class with a custom 'class' which has individual 'functions'.

While the ActionScript in Flash Lite 1.1 doesn't formally support Object Oriented (OO) programming it is still Object Oriented. Flash Lite 1.1 has all of the hallmarks of an OO language including inheritance, modularity, polymorphism, and encapsulation, but all in a non-formal way.

To utilize these OO techniques in an FL 1.1 application the developer must be familiar with the timeline in Flash and with the syntax of Flash Lite 1.1's version of ActionScript which is based on the version of ActionScript that was introduced in Flash 4 (the first version of Flash that had a scripting language) back in 1999. Furthermore the developer must choose a production methodology and design in the application to enable a more OO style of development.

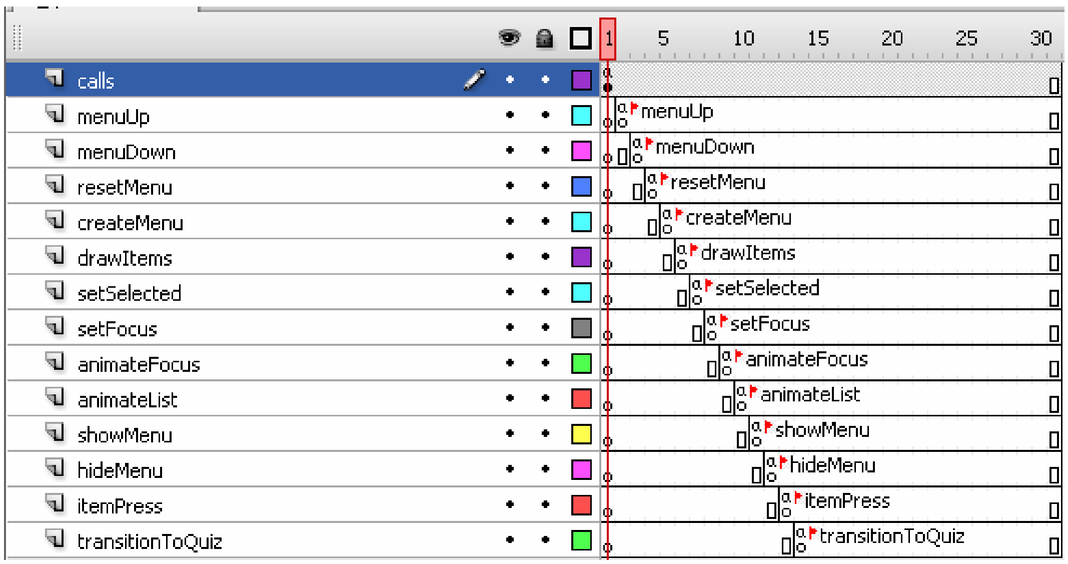
Below is a table that shows a mapping that can be applied to Flash Lite 1.1 projects.

Flash Lite 1.1 Object-Oriented Mapping	
ActionScript 2.0	**Flash Lite 1.1**
Class	Movieclip in the FLA with a child movieclip containing Keyframes named as functions.
Function	A named Keyframe with ActionScript - perhaps with only one #include statement
Classpath	A subdirectory named 'includes', 'classes', 'com', or similar – the example below shows a folder named 'com' which then contains other subdirectories of related AS files. 

Example: Keyframes as Functions

The image below shows the frame structure that can be used to effectively create 'functions' by naming the layers and Keyframes as you would functions in ActionScript 2 (AS 2.0).

By collecting all of the functions that would be declared in a class with ActionScript 2 in the same child movieclip it allows your syntax to be much closer to AS 2.0.



By naming the layers the same as the Keyframes you get the exact name of the 'function' in the Compiler Errors panel as the name of the layer. This makes it quite easy to debug your Flash Lite 1.1 project.

Example 'Location' in Compiler Errors panel:

ActionScript in a Keyframe:

```
Symbol=menuMethods, layer=menuUp, frame=2, Line 9
```

ActionScript in an included ActionScript file:

```
menuUp.as, line 9
```

Example #include statement for menuUp Keyframe:

```
#include "com/smashingideas/menu/menuUp.as"
```

Example call statement:

```
call("../menu:menuUp");
```

Equivalent AS 2.0:

```
_parent.menu.menuUp();
```

Link: Optimization Tips and Tricks for Flash Lite 2 (still a great reference for FL 1.1 developers)
by Jonathan Duran

http://www.adobe.com/devnet/devices/articles/optimization_tips.html

Link: Optimizing Flash Lite Content by Josh Ulm

http://weblogs.macromedia.com/xd/archives/2005/11/optimizing_flas.cfm

See Also: Further Reference

6

Related Editorial Content and White Papers

IDC White Paper: Creating a More Engaging Mobile Experience
http://www.adobe.com/mobile/platform/datasheet/idc_engaging_mobile_experiences.pdf
DVD: Total Training for Adobe Flash Lite 2.1: Creating Mobile Applications, by Dale Rankine
http://www.totaltraining.com/prod/adobe/flashlite21_cma.asp?c=adobe_center&n=developer_flash
Video: Total Training: Handling device key press events:
http://www.adobe.com/devnet/devices/articles/total_training_key_press_events.html
Book: Foundation Flash Applications for Mobile devices, by Richard Leggett, Scott Janousek, Weyert de Boer
http://www.friendsofed.com/book.html?isbn=1590595580
Book excerpt:
http://www.adobe.com/devnet/devices/articles/friends_of_ed_app_dev.html

Available Training

http://www.adobe.com/training/

Links

General Flash Lite Links

- Mobile and Devices Developer Center:
  http://www.macromedia.com/devnet/devices/

Flash Lite Updates

- Updates for Flash CS3 and Flash Professional 8– Mobile & Other
  http://www.adobe.com/support/flash/downloads.html

Flash Lite CDKs

- Flash Lite 1.1 and/or 2.x Content Development Kits (CDKs)
  http://www.adobe.com/devnet/devices/development_kits.html

Flash Lite: ActionScript Reference and Documentation

- Flash Lite 2.0 ActionScript Language Reference
  http://livedocs.adobe.com/labs/1/flashlite2_flash8updater/wwhelp/wwhimpl/js/html/wwhelp.htm?href=Part_api_ref.html
- Flash Lite 1.1 ActionScript Language Reference
  http://livedocs.adobe.com/flash/8/main/wwhelp/wwhimpl/js/html/wwhelp.htm?href=Part12_FL_API_Reference.html
- Flash Lite Documentation
  http://www.adobe.com/support/documentation/en/flashlite/

General Device Central Links

- Adobe Device Central CS3 Developer Center
  http://www.adobe.com/devnet/devices/device_central.html
- Adobe Device Central Online - device profile updates for Adobe Device Central
  http://www.adobe.com/go/dconline
- Device Central CS3 Documentation
  http://www.adobe.com/support/documentation/en/devicecentral/

Flash Lite Blogs & Communities

- Blog List at Adobe.com
  http://www.adobe.com/devnet/devices/index.html
- Flash Lite communities are listed on the 'Community' tab
  http://www.adobe.com/go/mobileusergroup

About the Author

Keldon Rush is the Director of the Platforms & Technologies Group at Smashing Ideas.
He has been creating software since 1995, using Flash since Version 2, and has been creating mobile software since 2000.

About Smashing Ideas

Smashing Ideas has been the leader in creating rich media and Flash-based entertainment for more than eleven years. Smashing Ideas develops brand-building experiences and engaging content for many of the top entertainment destinations on the web, and has created more than 300 games for multiple platforms including PC, IPTV and Mobile.

Smashing Ideas has offices in Seattle, (North American headquarters), Portland and in the UK. For more information, please visit www.smashingideas.com or www.smashingcontent.com .

Adobe

Adobe Systems Incorporated
345 Park Avenue
San Jose, CA 95110-2704
USA
www.adobe.com

95009544 6/07